BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

ADVANCED EVOLUTION OF CELLULAR AUTOMATA

POKROČILÁ EVOLUCE CELULÁRNÍCH AUTOMATŮ

HABILITATION THESIS
HABILITAČNÍ PRÁCE

AUTHOR **Ing. MICHAL BIDLO, Ph.D.**
AUTOR PRÁCE

BRNO 2022

Abstract

This work deals with the methods for advanced evolutionary design of cellular automata. Cellular automata represent a massively parallel computational model consisting of a lattice of locally interacting computing elements – cells whose states develop in time according to a set of transition rules which form a local transition function of the cells. Commonly the rules are represented as a table the rows of which determine new states of cells for given combinations of states in their neighborhoods. Since there is no general systematic approach how to design the transition rules, various (meta)heuristics have often been applied, including evolutionary algorithms. This work investigates the utilization of two novel representations of transition functions of cellular automata for their automatic design by means of evolutionary algorithms. Their detailed description and results are presented as a selection of papers published by the author which are included as appendices of this work. In particular, an instruction-based representation is proposed and its abilities evaluated on replication and pattern transformation benchmarks in two-dimensional cellular automata. The second representation, denominated Conditionally Matching Rules, which represents the main outcome of this research, is introduced and investigated in several studies. Specifically, it will be shown its utilization in the evolution of computational processes in cellular automata – the multiplication in binary two-dimensional automata and generic square calculations of natural numbers in multi-state one-dimensional automata. The crucial part of the studies of conditionally matching rules is devoted to the evolution of multi-state two-dimensional cellular automata. Advanced case studies are presented including the evolutionary design of replicating loops, pattern development problems and the utilization of conditionally matching rules for image filtering. Finally, a comparison of the conventional representation and conditionally matching rules was performed. It was demonstrated that the representation utilized for the evolution transition functions significantly influences the behavior that the resulting cellular automata can perform. In all proposed studies the conditionally matching rules enabled to obtain results which were not known before.

Abstrakt

Tato práce se zabývá metodami pokročilého evolučního návrhu celulárních automatů. Celulární automaty představují masivně paralelní výpočetní model sestávající z uspořádání lokálně interagujících výpočetních elementů – buněk, jejichž stavy se vyvíjejí v čase podle množiny přechodových pravidel utvářejících lokální přechodovou funkci těchto buněk. Obvykle jsou přechodová pravidla reprezentována jako tabulka, jejíž řádky určují nové stavy buněk pro dané kombinace stavů buněk v jejich okolí. Jelikož neexistuje obecný systematický postup pro návrh přechodových pravidel, jsou často aplikovány různé (meta)heuristiky, zahrnující též evoluční algoritmy. Tato práce zkoumá využití dvou nových reprezentačních technik přechodových funkcí celulárních automatů a jejich automatický návrh pomocí evolučních algoritmů. Jejich podrobný popis a výsledky jsou prezentovány jako soubor článků, publikovaných autorem této práce, které jsou zde zahrnuty v podobě příloh. Konkrétně je představena technika instrukční reprezentace přechodových funkcí, jejíž schopnosti jsou vyhodnoceny na problémech replikace a transformace vzorů ve dvourozměrných celulárních automatech. Druhá reprezentace, nazvaná podmíněně aplikovaná pravidla, představuje stěžejní část výzkumu této práce a tato je vyhodnocena na následujících případových studiích. Konkrétně je ukázáno její využití v evoluci postupů pro násobení v binárních dvourozměrných automatech a obecného výpočtu druhých mocnin přirozených čísel ve vícestavových jednorozměrných automatech. Hlavní část výzkumu podmíněně aplikovaných pravidel je zaměřena na evoluci vícestavových dvourozměrných celulárních automatů. Pokročilé případové studie zde zahrnují evoluční návrh replikujících se smyček, problém vývoje vzorů a uplatnění podmíněně aplikovaných pravidel pro filtraci obrazu. Nakonec je představena studie srovnávající tabulkovou reprezentaci a podmíněně aplikovaná pravidla. Je ukázáno, že použití konkrétní reprezentace pro evoluci přechodových funkcí zásadně ovlivňuje chování, které mohou výsledné celulární automaty vykonávat. Všechny zde zahrnuté studie prokázaly, že pomocí podmíněně aplikovaných pravidel je možné dospět k výsledkům, které dříve nebyly známy.

Keywords
Evolutionary algorithm, cellular automaton, transition function, conditional rule.

Klíčová slova
Evoluční algoritmus, celulární automat, přechodová funkce, pdmínkové pravidlo.

Reference
BIDLO, Michal. *Advanced Evolution of Cellular Automata.* Brno, 2022. Habilitation thesis. Brno University of Technology, Faculty of Information Technology.

Advanced Evolution of Cellular Automata

Declaration

Hereby I declare that this habilitation's thesis was prepared as an original author's work. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Michal Bidlo
October 12, 2022

Acknowledgements

The author would like to thank the colleagues from the Department of Computer Systems (UPSY) at the Faculty of Information Technology (FIT), primarily Lukáš Sekanina for maintaining excellent working conditions at the Department of Computer Systems of FIT, Zdeněk Vašíček for his help with preparing of the first studies of advanced evolutionary design and Jiří Jaroš for some practical comments and recommendations for this work. Thanks to PhD students, both from and outside UPSY, namely Marta Jaroš, Ondřej Olšák and Gabriela Nečasová for inspirative and supportine discussions during writing this thesis. Last but not least the author appreciates some other colleagues of FIT, primarily Petr Matoušek, František Zbořil jr., Filip Orság, Zbyněk Křivka, Jaroslav Rozman and Tomáš Vojnar for their willingness to share their ideas and experience which was also very helpful. A special thanks belongs to Jitka Soukupová who, even though has never seen me, supported my efforts indirectly during many discussions with my brother, by her discrete questions, curiosity about my work and encouraging messages.

Contents

1 Introduction **4**

1.1 Motivation 5
1.2 Goals 7
1.3 Thesis organization 8

2 Overview of CA Literature and Research Areas **9**

2.1 Identification and Synthesis of Transition Rules 9
2.2 Theoretical Aspects of CA 10
2.3 Cryptography and Security 11
2.4 Modeling and Simulation with CA 12
2.5 Modeling Crowds and Traffic Systems 12
2.6 Dynamics, Control and Synchronization 13
2.7 CA for Image Processing 14
2.8 Asynchronous CA 15

3 Evolutionary Approach to CA **16**

3.1 Evolution of binary CA 16
3.2 Evolution of Game of Life Structures 17
3.3 Evolution of multi-state CA 18
3.4 Recent studies of ECA 18
3.5 Intention of This Work 20

4 Advanced Evolution of Cellular Automata **21**

4.1 Instruction-Based Approach 21
4.2 Introduction of Conditionally Matching Rules 22
4.3 Evolving Computations in Binary 2D CA 22
4.4 Evolving Generic Computation in Multi-State CA 23
4.5 Evolution of Replicating Structures 23
4.6 Pattern Formation in CA Using CMRs 24
4.7 Impact of the Rule Encoding on the CA Behavior 24
4.8 Application of CMRs on Image Filtering 25

5 Conclusion **26**

5.1 Contributions 26
5.2 Main Outcomes 26
5.3 Future Work 28

Bibliography **29**

Appendicies – Paper Reprints **40**

I **Evolution of Cellular Automata Using Instruction-Based Approach** **41**

II **Evolution of Cellular Automata with Conditionally Matching Rules** **50**

III **Evolving Multiplication as Emergent Behavior in Cellular Automata Using Conditionally Matching Rules** **59**

IV **On Routine Evolution of Complex Cellular Automata** **68**

V **Evolution of Cellular Automata-Based Replicating Structures Exhibiting Unconventional Features** **82**

VI **Advances in the Evolution of Complex Cellular Automata** **105**

VII **Evolution of Cellular Automata Development Using Various Representations** **131**

VIII **Evolution of Cellular Automata with Conditionally Matching Rules for Image Filtering** **134**

List of Abbreviations

ASIC	Application-Specific Integrated Circuit
CA	Cellular Automaton
CMR	Conditionally Matching Rule
EA	Evolutionary Algorithm
ECA	Evolving Cellular Automaton
ES	Evolution Strategy
FPGA	Field Programmable Gate Array
GA	Genetic Algorithm
GoL	Game of Life
GP	Genetic Programming
NN	Neural Network
PRNG	Pseudo-Random Number Generator
PSO	Particle Swarm Optimization
VLSI	Very Large Scale Integration

Chapter 1

Introduction

Complex systems represent a wide interdisciplinary area in which various phenomena (e.g. natural, social, medical) are studied. Typically, a complex system consists of many *elements* (e.g. atoms, cells, creatures etc.) that *interact* in some sense. During these interactions, each element exhibits some properties which vary with time, can be observed, measured or detected and are usually referred to as a *state* of the element. Depending on the domain, an element my be, for instance, a neuron with synapses interconnecting other neurons, an artificial neuron whose outputs (analogically to synapses) express its "state" in the form of a value and allow transferring it to the inputs of other neurons, a particle whose state may express its spin together with its position and other properties.

The interactions of the elements with expressing their states during a time period represents a *behavior* or *development* of the system. The behavior of the elements, i.e. the changes of their states, are driven by some domain-specific *rules*. If one observes a single element, it is possible to determine its *individual* (local) behavior. The observation of the development of the system as a whole, i.e. the overall behavior of all its elements (visible or significant in some way), allows determining a *global* behavior of the system. For example, a diffusion of two liquids progressively changes the color of neighboring particles whose individual behavior is given by rules of the random Brownian motion in an observed area whilst the global behavior of the whole substance exhibits a process leading to the "stabilization" of the state of the elements which (when expressed by the color) finally appears to be the same for all the elements.

The goal of studying complex systems is typically to *analyze* the behavior observed in a given (existing) system and to understand its functioning, e.g. by deriving its mathematical model, the rules of how its elements interact. On the other hand, an opposite approach may be applied the goal of which is to achieve a specific (desired) behavior of the system by searching for (or selection of) a suitable arrangement of the elements, their functions and rules of interactions, i.e. to *design* or *synthesize* a (still unknown) system that ought to exhibit a specific behavior (i.e. to perform or solve a given task).

If a specific complex system is considered, the elements are usually well known entities for which a simulation model is available or can be derived and implemented together with the rules of interactions. Once a suitable implementation is available, the current technology allows in many cases *simulating* the behavior of a complex system as a whole (at least to some extent or a size of the system instance). This enables us, for instance, to perform analysis of its behavior, to make predictions, to identify unknown features, to tune or modify the system for new scenarios, all without the necessity to work with their real elements.

The simulation allows reducing the cost of studying the complex systems in general, simplifying substantially the process of analysis of complex systems in various conditions or even making these tasks possible in situations when working with real elements is dangerous or not realizable at all (e.g. the study of radioactivity or the research of the universe).

However, there are some major issues:

1. Although a precise model of the elements may exist, the rules of interactions may be known and the (individual) behavior of every single element may be observed (at least in some systems), the global system behavior cannot be simply derived from that. This feature is typical for complex systems and is referred to as *emergent behavior*. For example, cellular automata, regulatory networks or particle systems typically exhibit such behavior.

2. There is usually no central control that would determine the global system behavior – this behavior really emerges as a result of interactions of all the elements of the system and their individual behavior. Therefore, the manipulation (e.g. debugging, optimization, adaptation to various conditions) with such systems is usually not intuitive and traditional working paradigms may be useless. However, the emergent phenomena represents an important and interesting feature of complex systems as it may give arise to some processes (e.g. self-organization or replication), that could be difficult to achieve directly by a central control.

3. As a consequence of the previous paragraphs, there is another substantial issue regarding the study of complex systems: when a behavior or properties of a system ought to be determined for a future time period, it is usually necessary to perform a simulation of the entire system for a given initial state and conditions which may be highly time consuming. This issue becomes particularly serious in some systems whose number of elements may be enormously high (e.g. the number of molecules in a drop of water) which, in fact, makes a reasonable computer simulation practically impossible. Therefore simplifications are needed to be applied.

1.1 Motivation

This work will particularly deal with *cellular automata* as one of the sorts of complex systems. Cellular automata (CA) represent a massive parallel computational model composed of a regular structure of simple computing elements (*cells*), whose number is possible to scale up and arrange into multiple dimensions. Usually 1D or 2D cellular automata are considered in which the cells form a linear (circular) or rectangular (toroidal) grid, respectively. The state of each cell is typically expressed as an integer value (or color) and the states of all cells at every given moment is usually referred as a *configuration* of the CA, or sometimes a (global) state of the CA. There are only local interactions between the cells (i.e. each cell "sees" only the states of itself and its closest neighbors) and the development of the CA is performed in discrete time steps synchronously, i.e. in each step all cells acquire their new states at the same time. The calculation of the new cell states in each time step is performed by a *transition function* of each cell (referred to as the *local transition function*). This function is typically expressed as a set of *transition rules* specifying for each combination of states in the neighborhood of the given cell a new state of this cell. The global behavior of a CA is given by the transition functions of its cells, their interactions and the emergent phenomena.

The importance of the research in the area of complex systems and cellular automata in particular will be illustrated by several examples in the next paragraphs. Further, some fundamental issues related to the CA design will be mentioned on the basis of which a hypothesis of this work will be formulated. In general, the concept of cellular automata may be used in many areas, for both theoretical and real-world purposes.

In theoretical computer science, several CA instances were proven to be computationally universal. Perhaps the most simplest case is a 1D CA driven by the "famous" *Rule 110* as studied by Wolfram and later by Cook [32]. In 2D CA, however, probably the most known universal computing model is the Game of Life proposed by John Conway [45, 4] in 1970 (demonstrations of universality may be found, for example, in [57, 98, 99]). Another approach was proposed and proven by Sipper [116]. The aforementioned studies considered binary CA (i.e. those working with only two cell states, 0 and 1). A more advanced (4-state) and computationally more efficient computationally complete CA was proposed by Brian Silverman whose model is currently known as the Wireworld CA [36]. Although these CA represent theoretical benchmarks rather than really usable computing platforms, the issue of computational universality in homogeneous, massively parallel arrays may become important in the future, e.g. for the synthesis of nano-devices or quantum systems [56].

Probably the widest area where CA may be applied is that of modeling, simulation and prediction of the behavior of various natural as well as real-world systems. In this context, CA may provide a suitable research or application platform. For example, CA have shown as an efficient tool to study a wide range of biological systems, including the morphology of bacterial colonies [16], growth of the microbial biofilms [125] or recently the classification of the main variants of the SARS-CoV-2 virus [120]. More general studies consider various aspects of natural systems that could be beneficial in computers, in particular the replication, self-organization or adaptation [126]. Real-world applications of CA include, for instance, modeling urban changes [72], traffic microsimulation [65] or flood analysis [50]. In computer graphics, the concept of CA represents a basis for the raster image processing [103].

Basically the transition rules of CA are specified as a *table* the rows of which are of the form: $cell_1, cell_2, ..., cell_n \rightarrow cell_c^{'}$ where the part on the left of the arrow (the input of the transition function) represents states of cells in the neighborhood of a given cell and the right part (the output of the transition function) gives a new state of the cell (usually of that in the middle of the neighborhood) for the next time step. In more complex binary CA transition functions may consider the total number of active cells in the neighborhood as the primary input for calculating new states which are referred to as *totalistic* cellular automata [127] (e.g. the aforementioned Game of Life is a case of totalistic CA). However, the "programming" of a cellular automaton, i.e. the design of suitable transition function in order to achieve a given global behavior, represents a difficult task for which there is no universal approach. Although the synthesis of transition rules was approached from various perspectives including, for instance, evolutionary computation [79], this task still remains the biggest challenge for routine cellular automata applications. Therefore, there is an effort to automate and optimize this process and to study new methods of how to represent the transition rules in order to make this task realizable using current technologies.

So far, some specific cellular automata have been designed "ad hoc" for several, mostly benchmark or artificial, problems. Probably the most known instances include Langton's [70] or Byl's [26] self-replicating loops, CA simulating artificial life [71] or various structures designed for specific purposes in the Game of Life CA (e.g. see [98, 111, 45]). Nevertheless,

such approach to discovering the CA rules requires an experienced designer and becomes intractable for complex non-binary (i.e. multi-state) CA with bigger cellular neighborhood.

The process of automatic CA synthesis for a given target behavior usually leads to *searching* for a suitable set of rules in huge spaces of transition functions induced by the given CA type (the cardinality of the search space is primarily determined by the CA dimension, the number of cell states and the size of the cellular neighborhood; it grows exponentially for increasing values of these parameters). Although the simplest cases of 1D binary CA may be analyzed and studied by means of exhaustive methods (as performed by Wolfram [127]), more complex instances and applications still represent a significant obstacle the solution of which, in fact, *requires* some advanced kind of the search automation. From this point of view evolutionary algorithms may provide robust searching mechanisms as indicated (even though for binary CA only) by Cenek and Mitchell in [29].

Based on the aforementioned statements, the following hypothesis may be formulated for the purposes of this thesis. The hypothesis indicates the overall research objective of this work the goal of which is to demonstrate further abilities of the evolutionary approach for the automatic synthesis of the cellular automata rules:

By introducing a suitable form of transition rules it is possible to significantly improve the evolutionary design process of cellular automata with a remarkable influence on the potential tasks for which the rules can be obtained as well as on the behavior of resulting CA themselves.

1.2 Goals

The research conducted within this thesis represents an experimental work regarding the area of cellular automata which represents the main research interest of the author for more than 10 years. During this period more than ten original papers have been published regarding the *evolutionary design* of CA rules in various domains. Among the firsts, the CA were applied to generate logic circuits, to perform replication of selected structures or to calculate some arithmetic operations directly in the cellular array. These experiments showed that the selection of a suitable encoding of the CA rules (i.e. the form of the transition function) represents a key issue for a successful evolution in order to achieve a given CA behavior since the traditional table-based representation showed to be insufficient. The evolution exhibited low success rates or even was not able to find any acceptable solution.

The subsequent research was focused on improving the representation techniques together with tuning the evolution in order to find successful results for complex non-binary CA as well (e.g. those working with 8 cell states or more). In particular, the work included the comparison of the basic representation with a more advanced instruction-based encoding and, further, an encoding in a form of so-called Conditionally Matching Rules (CMRs), representing probably the best so far method for the evolution of multi-state CA. In general, the conducted experiments showed that the new encoding of transition functions not only allows optimizing the evolutionary search process but also provides results (in the form of the CA behavior) that have never been achieved before. Moreover, a comparison of the results obtained by means of the table-based encoding and the Conditionally Matching Rules showed a significant influence on the behavior of the CA itself in solving the same problem.

On the basis of the observations mentioned above, the following partial objectives of this work may be formulated:

1. To introduce an instruction-based representation of transition functions and demonstrate its ability on the evolution of simple replicating structures.

2. To introduce a concept of Conditionally Matching Rules which represents a fundamental idea of the author and constitutes a major research outcome of this work. To demonstrate its abilities in solving basic replication and pattern transformation tasks.

3. To demonstrate the CMR concept in solving some advanced benchmark problems; in particular, the replication, generic square calculation and development of stable structures from a seed will be investigated.

4. To compare the features of resulting CA obtained by means of table-based method and the CMR approach; in particular, it will be shown that both methods provide totally different CA behavior in solving the same tasks and only the CMR method was able to develop stable structures in the cellular array.

5. To demonstrate the application of Conditionally Matching Rules in the task of designing image filters; specifically, CA with 256 cell states will be applied on removing salt-and-pepper noise of the intensity up to 80%.

1.3 Thesis organization

The text is composed of a collection of relevant conference and journal papers to which the following chapters provide an overall comment. Specifically, chapter 2 summarizes recent work in the research and applications of cellular automata and identifies some remarkable well-established areas where current studies related to CA take place. Chapter 3 provides a similar overview with the focus on the application of evolutionary algorithms in combination with CA. On the basis of this summary the orientation of the research presented in this thesis is given in Section 3.5. Chapter 4 describes the main research ideas of this thesis together with highlighting important results and provides references to relevant papers included in Appendices of this work. The thesis concludes by Chapter 5 where an overall summary is presented together with a brief discussion about possible future work.

The main part of this thesis — the research outcomes in the form of a collection of relevant papers published by the author — is presented in Appendices starting by page 40.

Chapter 2

Overview of CA Literature and Research Areas

In this chapter a summary of well-established CA research and application fields is provided together with a selection of state-of-the-art papers published during recent years. In addition to the areas surveyed in more detail in the subsequent sections, some other domains could be included as well (e.g. CA-based hardware, multi-agent systems and networks) whose focus is, nevertheless, more distant from the research presented in this thesis. However, recent studies from these areas may be found, for example, in [51, 78, 129, 128, 118].

In addition to the survey presented in this chapter, a separate overview of *evolutionary algorithms* applied in combination with cellular automata will be proposed in Chapter 3. On the basis of this summary and the results obtained so far in this area, a particular intention of this thesis (i.e. the orientation of the research) will be described in Section 3.5.

2.1 Identification and Synthesis of Transition Rules

The CA behavior is represented by a sequence of configurations (states of each cell) varying in time. At every time step the next configuration is determined by the local transition function of the cells. Note that in this text only *uniform* CA will be considered, i.e. there is a single transition function determining the behavior of all cells. However, the task of designing a CA is to find this transition function which will perform the behavior of interest of the CA or a transformation (in general in several time steps) of an initial configuration onto a target configuration. Depending on a particular scenario, the process of discovering the transition function may be, in principle, considered either as *identification* of transition rules or *synthesis* (i.e. the design) of the rules.

The **identification** of the transition rules assumes that a sequence of configurations (i.e. the CA development for a finite number of steps) is known for every time step that the CA is required to pass through. The task is to identify the transition rules according to which the cells update their states in each successive step. The solution of this problem is fundamentally based on the method proposed by Adamatzky in [3]. His approach was later improved, e.g. by combining it with Learning Classifier Systems [24] or advanced representation techniques like polynomial representation or decision trees [5]. Other approaches to this issue, e.g. such based on incomplete observations, neighborhood detection or combinations of these principles with evolutionary methods, have also been published [132, 75, 22, 23]. This thesis will not deal with the identification scenarios.

A more general task is to perform a **synthesis** of the transition rules for cases where the exact CA behavior is not known. In this scenario the design of the transition function needs to be, in fact, performed together with the discovery of (at least a part of) the CA development that fulfills the given requirements (for example, to achieve a specific configuration from a particular initial configuration, to replicate a given structure, to perform a periodic process, to achieve a stable configuration, etc.). Usually the number of steps of the CA needed to perform the given task is not exactly known and even the existence of a valid solution for given CA parameters cannot be guaranteed. **The task of rule synthesis will be considered in this thesis and evolutionary approach will be applied for its solution**.

The following sections contain an overview of some important research and application areas in which CA have been studied together with mentioning selected techniques applied to the solution of particular problems. As there is no unified approach to the synthesis of the CA rules, it is not possible to provide an exhaustive list of methods which are typically application-specific.

2.2 Theoretical Aspects of CA

Since the first extensive studies of cellular automata, performed by von Neumann ([90], first published in 1958, and [89]), and particularly after the publication of Conway's concept of the "Life in a grid" in 1970 [45], researchers started to think about CA in a wider perspective. Nevertheless, besides various application areas that have emerged during the time, theoretical foundations of CA with some closely related subjects like computation, complexity or implementation issues, still remain important research domains.

For example, an interesting CA-based method was recently published considering the computation of shortest paths in grid graphs. 2D cellular automata were applied where grid graphs are represented as configurations of these CA with nodes and edges modeled by cells with different state sets. The authors discovered that the worst case time complexity is $O(n)$ where n is the number of nodes of the connected graph [15].

CA are often investigated towards their ability to accept formal languages in order to determine their computational capacity. Kutrib and Malcher proposed an opposite approach by looking at CA towards their ability to *generate* formal languages. The authors considered 1D CA and demonstrated their capability to generate the Thue-Morse sequence within real time and the generation of unary patterns in depth and obtaining a characterization by time-constructible functions and their corresponding unary formal languages [66].

From both theoretical and application perspective, maximal length n-cell CA are of a high interest (e.g. as pseudo-random number generators, VLSI testing, cryptosystems etc.). Such CA have the cycle of the length $2^n - 1$. In [2] the authors investigated non-linear maximal length CA and proposed a method for their synthesis. A similar issue was studied also in [47] together with a FPGA-based implementation. It is worth noting that a more general investigation regarding the maximum length cycles in CA was proposed in [59] where composited cellular automata were considered.

There are so-called partitioned CA constituting a subclass of standard CA in which each cell is divided into several parts, and the next state of a cell is determined only by the adjacent parts of its neighbors. For example, this concept may be used for designing reversible CA. Morita investigated a class of partitioned CA (particularly so-called elemen-

tary triangular partitioned CA) and discovered a new instance that is capable of universal computation [85].

In addition to the already mentioned rule 110, which is able to perform universal computation, the issue of universality has also been studied in [53] in CA with the rule 184. The authors showed how such CA can implement any logic circuit by demonstrating a possibility to construct a computationally complete set of basic logic gates. Finally an implementation of the adder circuit was proposed in this paper using the CA with rule 184. Another (a more specific) study of universality was published in [60] regarding the asynchronous brownian CA requiring merely two transition rules and three states.

The elementary binary 1D CA have often been studied theoretically in order to determine their capabilities and properties where a specific rule has been considered. Rules 110 and 184 have already been mentioned above. In addition to that, the computationally universal rule 110 has also been studied from the point of view of spectral properties of the computation process [91]. The authors suggest a relationship between a noise which is possible to identify and quantify from the CA and computability of such CA. Another rule, particularly the rule 20, was examined in [92] with respect to so-called conflict-like dynamics of the appropriate CA suggesting that this rule is composed of two simpler CA (the authors also mention rule 14 with similar property).

2.3 Cryptography and Security

The simple concept of elementary CA has lead to their applications in the area of error detecting and correcting codes and cryptography, often with a support of hardware implementations. For example, the proposal of a single-byte error correcting and double-byte error detecting code published in [31] represented a remarkably simple and cost effective alternative to the existing Reed-Solomon codes available at that time. Recently, this issue was revisited and a CA-based single-byte correcting code was adapted for memory systems [109]. The authors proposed several variants of the code and performed simulations of their implementation in modern FPGA and ASICs. The results showed a possibility to reduce the hardware demands substantially in comparison with the existing Reed-Solomon solutions. A generalized solution considering 3-byte and 4-byte error corrections was published in [17]. According to the mentioned studies, in many cases the CA-based error correcting schemes have shown as simpler and cheaper alternatives to the appropriate variants of Reed-Solomon coders.

Another area in which CA proved their capabilities is cryptography. For instance, semi-bent or bent Boolean functions are interesting from a cryptographic point of view, since they possess several desirable properties which are useful to resist linear cryptanalysis. Recently, CA have been investigated with respect to their ability to generate such funtions [77, 44]. In addition, a new method utilizing Genetic Algorithms (GA) to evolve 2D CA as pseudo-random number generators (PRNGs) was proposed in [48] where the authors introduced a composite fitness metric which allows quantifying the performance of resulting CA with respect to individual tests, i.e. it allows to find the CAs that are most likely to pass among a set that fails a specific PRNG test.

From a more general point of view, CA have evolved as a good cryptographic primitive during recent years for solving various (sub)tasks in this domain. For example, PRNGs represent important components of stream ciphers and CA have shown capable to combine some partial results attained in this area with existing traditional stream ciphers in order to provide new systems with exhibit better cryptographic properties [62, 68, 63, 34]).

2.4 Modeling and Simulation with CA

The development of modern computing technology allowed overcoming some demands that struggled to perform efficient calculations with CA-based systems or to apply the CA to simulate and analyze real-world or natural complex systems. For example, the latest FPGA technology provided a powerful platform to accelerate the computation of CA or massively parallel systems in general (e.g. as recently demonstrated in [67, 12]). In addition to FPGAs, modern graphic processing units (GPUs) currently allow performing general-purpose computations and speed-up significantly some kinds of tasks. The utilization of GPUs for CA was recently proposed in [28] and in combination with evolutionary techniques for generating CA rules in [123]. However, some attempts have also been performed to implement CA by means of future technology (e.g. using graphene nanoribbons [96]).

The application of CA to model and simulate real-world systems include many areas from which the following were published in the last years. In [1] a new method for evaluating the characteristics of photovoltaic panels was proposed. The authors introduced a two-component approach integrated in a CA-based model for processing the evolution and distribution of the temperature and the electrical output characteristics of the solar cells. The combination of results from these components then allows simulating the behavior of the panel numerically.

Modeling the wind flow and fire spreading belongs to complex natural phenomena for which CA were applied in the past. New studies allowed improving some existing models and increase their accuracy. For example, an advanced wind flow model and its impact on the forest fire spread was proposed in [61]. The authors integrated rules not only the wind direction and speed but also an improved forest fire model and other physical and climatic attributes into the CA (e.g. topography, land use, nature and density of vegetation or humidity). Another approach considering a discretized 3D cellular model for modeling the wind flow was recently proposed in [25].

As regards the simulation and CA rather from theoretical perspective, it was shown in [64] that for any non-uniform CA (i.e. such that considers a separate transition function for each cell), there exists a uniform CA that can simulate the non-uniform one. In order to do that, a function is introduced which maps the rules and states of a non-uniform CA to the states of the proposed uniform CA. The authors show that different applications that use different non-uniform CAs can now be implemented in one system, and also propose a brief overview of such applications. Another study, considering the simulation of CA by means of another computational model was published in [130] where the author constructs a series of infinite Petri nets which directly simulate the elementary cellular automaton Rule 110 (more variants of this model were presented).

2.5 Modeling Crowds and Traffic Systems

The simulation of spatial behavior during time belongs to the problems often considered in relation with CA. Modeling the crowd dynamics, pedestrian movement or traffic flow (as a more specific subclass of the general area related to the modeling and simulation using CA) represent typical examples of real-world complex systems. During the last years CA have shown to be a simple and effective platform for this kind of problems.

The CA-based pedestrian simulation models represent interesting alternatives to particle systems employing a continuous spatial representation. In CA, however, the space is discretized which may imply difficulties in modeling real phenomena (e.g. heterogeneity in

the walking speed of pedestrians in urban areas). This aspect was studied in [13] where an adaptation of this concept was proposed for managing the heterogeneous speed profiles in pedestrian models. In particular, in the discrete space of the CA, the authors work with a maximum speed of one cell per time step, but model lower speeds by having pedestrians yielding their movement in several turns, allowing to consider various sorts of pedestrians walking by different speed. A subsequent study reports the results of controlled experiments performed with various crowd and pedestrian models considering a personal space in static and dynamic situations: the area surrounding human body, linked to crowding due to spatial intrusion/restriction. Simulation results together with the parametric evaluation of pedestrians' psychological stress reaction to density were presented in [49].

Another study, considering a CA-based model for modeling crowd behavior management in airports, was published in [81]. The crowd dynamics is typically studies in specific situations whose conditions and parameters must be taken into account when designing the CA model (this is particularly true for various airport areas which was the subject of the aforementioned study). Another scenario was recently published studying spatial games and memory effects on crowd evacuation behavior by means of CA [82]. It is worth noting, however, that the pedestrian or crowd behavior is typically not deterministic. An enhanced CA model, called the fuzzy cellular automaton, which is able to overcame the limitations of standard CA models, was proposed in [46] where a GPU acceleration of the evaluation of the fuzzy CA was also applied.

An important part of this domain, with its own specificity, is the issue of modeling and simulation of road traffic situations (including crossings of different forms, multi-lane and highway traffic etc.). The concept of cellular automata has shown to provide a powerful tool to solve these tasks which may be considered on both the macroscopic o microscopic level. For example, a task of calibration of traffic microsimulation models was studied in [65]. The authors proposed a Genetic Algorithm-based approach with self-adaptation for tuning the CA parameters which allowed to obtain more accurate results. More specific traffic scenarios have also been studies recently (e.g. a cellular automaton models tuned specifically for multi-lane traffic, including lane-changing activities [131, 33]).

2.6 Dynamics, Control and Synchronization

CA as dynamical systems were, probably for the first time, formalized in [55], an overview of some modern approaches may be found in [35]. Nowadays the subsequent research gave rise a new subfield of CA where various tools (mathematical, statistical, experimental etc.) are applied in order to discover new features of the CA development. The studies include not only the processes (changes, dynamics) that can be observed in "patterns" generated by CA but also various issues regarding the control, synchronization, computation and extensions of the standard CA model.

For example, probabilistic CA have shown as a tool suitable to model many natural phenomena. Some of main control problems of probabilistic CA are reachability and drivability. The first is related to the possibility of applying a suitable control able to make the system reach a given state or a set of states. The drivability problem may be considered somehow complementary to the reachability one. The regional control of probabilistic CA was, in relation with the mentioned issues, studied in [11, 9] and a related local synchronization issue in [10].

One of the benchmarks often investigated in CA is the synchronization task which tries to achieve a periodically alternating homogeneous global configurations from given

(or arbitrary) initial configuration. Since the cells (in the standard CA) interact only locally, the solution of this task is non-trivial. One of the advanced study dealing with the synchronization task was published in [101].

Another phenomenon studied in the CA dynamics is the issue of attractors. An attractor may be viewed as a (final) configuration (or somehow restricted sequence of configurations) of the CA that the CA achieves from a given (or arbitrary) initial configuration. A point (or single length cycle) attractor is such an attractor which the CA reaches after only one step from the initial configuration. A new approach for the synthesis of non-uniform CA having only point attractors was proposed in [86].

In elementary cellular automata the new configuration depends solely on the configuration at the preceding time step. However, this concept may be extended by introducing a memory which allows the CA to develop in such a way that the new configuration depends not only on the current one but also on a certain summary of past configurations. This concept may be beneficial for modeling some kinds of systems exhibiting so-called non-markovian phenomena whilst the mapping defining the transition rules of the system (i.e. the formal model) remains unaltered. Dynamical systems with memory (including cellular automata) were summarized in [7]

2.7 CA for Image Processing

The concept of local cell interactions in 2D CA actually copies the same principle utilized for image processing. Therefore, CA with various modifications have often been considered to design image operators like noise removers, edge detectors and others [104, 102].

For instance, a new scheme using 2D CA for an image encryption was proposed in [30]. The authors discovered that the random evolution of CA along with masking makes the scheme secure and efficient in parallel software implementation. A security analysis has shown that the scheme with high diffusion and confusion is strong against correlation and differential attacks.

Several studies have recently dealt with edge detection by means of CA. In [42] the authors explored the possibility of using high-order CA (a variant of CA which can memorize a certain finite number of its past states) to perform edge detection. Experiments were devoted to show how to find optimal parameter values for the proposed model and the results showed to be very close to the best performing commonly used methods. Another approach was proposed to design a CA-based edge detector adapted to the particularities of the image where Genetic Algorithm was applied to identify the best CA rules, which was considered as an optimization problem. The authors claim that some weak points of a well-known Canny detector can be overcome by this method [38]. An interesting method was published in the past considering 2D 256-state CA where the gene expression programming was employed as the learning algorithm for edge detection in which the chromosome encodes the transition rule as the expression [115]. Another method was recently focused on biomedical images [14].

Image reconstruction and denoising represents an important non-trivial task of image processing for which CA have also been applied. Salt and pepper noise filtering by means of fuzzy-CA was studied in [108] where a local fuzzy transition rule is examined which gives a membership value to the corrupted pixel neighborhood and assigns next state value as a central pixel value. The authors showed a possibility to removes the noise effectively even at noise level as high as 90%. A more specialized method investigated denoising of biometric images: In [121] a new technique called an Iterative Refined Noisy Pixel Restoration was

proposed in combination with CA to get rid of salt-and-pepper noise, assessing the output image quality by means of various metrics and providing better performance compared to the alternative approaches. Finally, let us mention a method, inspired in physics, called fractional integral function, which was used for representing cellular automata model and cell information memory vector to perform face and fingerprint recognition [52].

2.8 Asynchronous CA

Asynchronous CA, nowadays studied as a distinguished subfield of cellular automata, represent a class of CA in which each cell may be updates to its next state an unbounded number of times according to a locally discrete time. This means that the updates may be deterministic, non-deterministic, random, sequential or even synchronous [88]. It was shown that for each synchronous CA an equivalent (i.e. exhibiting the same behavior) asynchronous CA may be constructed [87]. Recent surveys are available e.g. in [40, 41]. More specific studies on asynchronous CA include the issue of reversibility [114] or embedding the rules of the Game of Life into the asynchronous model where also a delay-insensitive circuit was proposed for its implementation [133]. Recently, asynchronous CA were studied with respect to the solution of the synchronization benchmark in 1D binary CA with a proof of correctness of the solution and determination of the upper-bound on the convergence time to the solution [107]. A study of recurrent rules, their classification and utilization to hiding some configurations of CA under fully asynchronous updating scheme was proposed in [106]. Asynchronous CA also showed as a suitable tool to enhance the quality of some pseudo-random number generators when identifying the CA of full-length cycle [105].

Chapter 3

Evolutionary Approach to CA

The simulation of CA-based systems on classical computers represents a computationally demanding task where (typically) thousands of cells need to be updated during each step of the CA in order to determine its behavior. The application of evolutionary algorithms (EAs) for tuning or synthesis of the CA rules was enabled by the development of the computing technology since the second half of 90s when the first successful studies were published mostly dealing with some non-trivial benchmark problems in binary CA (e.g. density classification or synchronization task). Evolutionary algorithms have subsequently been widely applied in combination with CA since there is no general systematic way to solve the synthesis of CA rules effectively. Note however that the evolutionary approach to CA represent a challenging issue even for the modern computers, hence sometimes there is an effort to accelerate this process on a dedicated parallel HW (e.g. using multi-processors and GPUs [123, 124, 27, 28]).

This chapter briefly summarizes the history of evolutionary algorithms applied on cellular automata (which have later been referred as *evolving cellular automata* – ECA), identifies some distinguished sub-fields and highlights important results. On the basis of the development of ECA the orientation of the research of this thesis is described in Section 3.5.

3.1 Evolution of binary CA

One of the first serious study of ECA was proposed in [8] when the majority (or density) classification benchmark was considered in binary 1D CA. Despite the fact this problem cannot be solved perfectly in 2-state CA [69], it represents an optimization benchmark suitable for investigating the behavior of complex systems and algorithms for their tuning. The authors applied Genetic Programming with automatically defined functions to find a suitable transition function that provided the accuracy of the classification 82.326% which was the best result compared to all known solutions at that time. To solve this task, the CA should stabilize into all-1s configuration if the number of 1s in the initial configuration exceeds the number of 0s and into all-0s otherwise. The solution is challenging since the decision relates to global CA feature but the CA development (calculating the decision) is based only on local cell interactions.

A review of a wider research from 1993-1996 related to evolving computations in CA by means of Genetic Algorithms was published in [80]. In addition to the majority classification, the synchronization task and issues related to the possibility of performing computa-

tions in locally interconnected "processors" were addressed. This research may probably be considered as the first application of evolutionary techniques to obtain a useful (although still rather at a benchmark level) emergent computation in CA.

Sipper et al. was among the firsts who tried to introduce a general evolutionary method for designing *non-uniform* CA. The algorithm was denoted as *evolutionary programming* and represented, in fact, a parallel EA that evolves local transition rules derived just from neighboring cells on the basis of the evaluation a desired (global) CA behavior. A hardware architecture was also proposed for this approach [117, 116].

It is worth mentioning that many other works from this period (e.g. those from Wolfram, Langton, Packard, Ruppin, Bäck, Juillé or Pollack), investigating similar problems, were remarkable. Also note that this research was mainly focused on binary 1D or 2D CA which represent the simplest and fundamental setup.

3.2 Evolution of Game of Life Structures

The search for particular CA rules for a given CA behavior is not, however, the only way how to evolve the CA. If a suitable CA is known (i.e. for which the transition function is available), then a target behavior may be designed by searching for a suitable (initial) configuration of the CA. This approach was applied several times for the Game of Life (GoL) CA.

Gliders and glider guns represent complex structures with specific cyclic behavior in the GoL CA. They were also utilized to prove the universality of the GoL CA. Therefore these structures are interesting from a more general perspective and the main issues are how they can be discovered, what kind of different variants may exist, what are their properties and whether it is possible to use them in more complex applications.

The evolutionary search (by means of Genetic Algorithm) for glider guns in GoL CA was introduced in [112] and revisited in [113] with a comparison of various techniques for their discovery. In fact it was shown that the original Gosper's glider gun (which is the most known in the GoL CA) just represents one of the possible structures that is capable of periodically generating other structures. Extended studies of this topic were presented in [110, 111] where the authors also introduced a classification of glider guns that takes into account the number of emitted gliders of a specific type.

An independent research of selected GoL-related structures was presented in [6]. In addition to gliders, the authors also considered other structures like R-pentominoes or exploders. This study had contributions in several aspects: (a) one of the objectives was to maximize the number of the structures of interest emerging in the CA and (b) several similar GoL CA were investigated (i.e. not only the Conway's original concept).

Although not directly related to evolutionary approach and Conway's version of the Game of Life, it is worth mentioning a recent study of glider guns published in [119] where the authors presented a new 2D binary CA with so-called *Sayab rule* which allows constructing a glider gun with just four live cells at its minimal phases – this is probably the most simplest variant of glider gun known at that time. It was also shown that this glider gun can implement complex dynamical interactions and gates required for computational universality [119].

Probably currently the most recent study of the evolution of the GoL-related CA was published in [122]. The author studied the evolution of autopoietic structures (i.e. "seed" objects in the GoL array that exhibit some specific behavior like stability, oscillations, movement etc.) using a variety of biologically analogous aspects integrated into a so-

called S-model together with an advanced fitness evaluation of the evolving autopoietic structures. Finally it is claimed that successful seed patterns are those that create a diversity of autopoietic structures.

3.3 Evolution of multi-state CA

Discovering of suitable set of rules for multi-state (i.e. non-binary, working with more than two cell states) CA represents a challenging task because the number of possible transition functions grows exponentially depending on the number of states as well as on the size of the cellular neighborhood. Therefore the utilization of EA in this case is fully justifiable.

The 3-state CA may be considered as the most simplest models in this category with which an interesting study was presented in [95]. The authors applied Genetic Algorithm for the evolution of rules of 3-state 2D CA the goal of which was to solve a *binary* classification problem in the 2D array as described in [58]. It was shown that the best found rules perform better than the manually designed heuristic CA rule and also outperforms one of the most widely used statistical method – the k-nearest neighbors algorithm.

The replication problem represents a typical benchmark related to multi-satate CA working with more than 3 states. In addition to the pioneering work of von Neumann with self-replicating machines and self-replicating loops for which the rules were designed manually (e.g. those from Langton or Byl), researchers started to investigate the abilities of EA for automatic discovery of self-replicating structures. Probably the first successful application of Genetic Algorithms to design rules for replicating structures in CA was published in [73, 74]. The authors used a linear encoding of elementary transition rules and introduced a new concept of CA called Effector Automaton (EfA) which allowed them to discover new replicating mechanisms that differ substantially from the manually designed solutions.

An advanced method for the evolutionary discovery of replicating structures in CA utilized Genetic Programming (GP) and provided an improved performance against the aforementioned methods (the original idea and initial results were published in [93]). The authors applied a tree-based representation naturally supported by GP and introduced new structures called S-tree to encode a *seed*, i.e. an object representing the initial state of the CA from which the replicas ought to be generated, and R-tree representing the encoding of CA rules for the evolution by means of GP. This approach enabled to evolve rules for a variety of structures to be replicated as well as mechanisms how the replication may be performed [94].

3.4 Recent studies of ECA

In addition to the studies mentioned in the previous sections, there has been a research of ECA during recent years from which it is worth mentioning some remarkable articles.

For example, in 2011, probably one of the first studies combining CA and Neural Networks (NN) was published in [37]. The authors investigated the possibility to generate complex self-organizing structures in CA whose cell behavior is controlled by NN integrated in them. EA was applied to train the NN in order to achieve the desired behavior. Although for complex patterns represented by the CA (e.g. paintings or photographs) the output is only a rough approximation of the overall mean color scheme, the idea of combining CA and NN introduced an interesting concept that was later revisited in a modified

form with remarkable results. In [84] the authors introduced a concept of Growing Neural Cellular Automata for the investigation of self-organization and morphogenesis in multicellular organisms. Their approach used *continuous* state values which allowed to design differentiable transition rules for better control of various phases of morphogenesis observed in natural systems. However, such rules were not invariant to rotation of the artifacts in the cellular space which represents a limitation prohibiting the existence of differently oriented instances of the target pattern on the same grid. Therefore an improved concept was introduced and denominated Isotropic Neural CA [83] which demonstrated an ability to grow accurate asymmetrical patterns through several methods with an advanced ability of the patterns to self-repair. Another NN-based concept was applied in [39] where an evolutionary algorithm was used for producing spiking neural systems in CA that emulate the patterns of behavior of biological neurons in vitro.

Another concept of CA with continuous state values, denominated *MergeLife*, was introduced in [54] representing a Genetic Algorithm which is capable to generate full color (with 16-bit color depth) dynamic animations according to aesthetic user specifications. The authors proposed several novel fitness measures that when given human selected aesthetic guidelines encourage the evolution of complex long-running animations that often include spaceships, oscillators, still life objects or even Universal Turing Machines.

Although Genetic Algorithm has often been applied to search for the CA rules, it is not the only EA which may be suitable for this task. However, a wider systematic study of different EAs in the areas of cellular automata is missing mainly due to a strong dependence of their applicability on the particular problem to be solved. Nevertheless, it is worth mentioning a comparative study of Genetic Algorithms and Particle Swarm Optimization methods (PSO), that was recently presented in [100], where the author also proposed a new PSO variant called Binary Global-Local PSO (BGL-PSO). Whilst the GA generally outperformed the PSO methods in both case studied (the density classification problem and a newly proposed generation of "chaotic" CA), there are some interesting observations concluding the paper and highlighting some features in favor of PSO [100]:

> "While PSO was — absent aggressive non-local jumping — more prone to be caught in local extrema, many times these local extrema were nonetheless found relatively early in the search with PSO; moreover, although the GA was typically more effective at finding a single best candidate solution, the PSO algorithm by contrast commonly found more instances of different quality solutions. Our BGL-PSO algorithm also consistently outperformed the binary PSO algorithm for the given CA-related tasks."

Evolutionary Strategies (ES) were considered for adjusting epidemiological model of Chagas Disease based on CA [43]. The authors considered an existing model previously adjusted by GA whose performance, however, declined with the expansion of the search space. Therefore an improved method was introduced and tuned applying a new multistage ES, where different settings are applied based on the current stage of the evolutionary search. The authors concludes their work by highlighting a fact that the proposed ES provided solutions with the least error in the set of experiments, demonstrating the improvement over the previous approach.

To conclude this section, let us mention a recent study investigating the application of EAs for designing reversible cellular automata, particularly those whose local update rules are defined by conserved landscapes [76]. The authors compared GA and GP when formulating this task as an optimization problem and approached its solution in three ways through a single-objective, a multi-objective, and a lexicographic optimization. Again, their

analysis demonstrates some interesting properties of those EAs applied on this problem. Le us highlight the most important observations [76]:

> "In the single-objective approach, we observe that GP can already find an optimal solution in the initial population. This indicates that evolutionary algorithms are not needed when evolving only the reversibility of such CA, and a more efficient method is to generate at random syntactic trees that define the local update rule. On the other hand, GA and GP proved to be quite effective in the multi-objective and lexicographic approach to (1) discover a trade-off between the reversibility and the Hamming weight of conserved landscape rules, and (2) observe that conserved landscape CA cannot be used in symmetric cryptography because their Hamming weight (and thus their nonlinearity) is too low."

3.5 Intention of This Work

As evident from the previous sections, the research of CA is (practically always) performed for a specific application or benchmark problem for which suitable CA rules ought to be found. Even though a more general technique is proposed for their synthesis, representation or computation (e.g. Cellular Programming [116] or Neural Cellular Automata [84]), the final method must be tuned for the presented case studies. The same approach will be considered in this thesis.

Various remarkable concepts have been introduced in order to improve the traditional CA design, its efficiency or suitability to solve particular tasks (e.g. asynchronous control, continuous state values, non-rectangular grids, effector automata, special rule encoding for a given problem, integrating other complex models into the cells – for example Neural Networks, etc.). In this thesis, however, only the basic concept of CA is considered and the main objective is to make the process of evolutionary CA design more efficient and accessible for multi-state CA through a new representation of the transition functions in the form of so-called *Conditionally Matching Rules* (CMRs).

The proposed approach was motivated by the simplicity of the standard CA concept and the elementary representation of its transition functions by means of a *table of transition rules*. The CMR encoding, in fact, generalizes the table-based method, preserving its simple concept and allowing to be converted uniquely back to this elementary form. Since the CMRs showed promising qualities in the initial study, it later became a basis for a wider research regarding the evolutionary design of multi-state CA solving various non-trivial problems and working with up to 256 states. This research constitutes the main part of this thesis with the details and results being presented as a selection of papers, published by the author, in Appendices I-VIII.

Chapter 4

Advanced Evolution of Cellular Automata

This chapter introduces the main research ideas of this thesis together with highlighting important results and providing references to relevant papers included in the Appendices.

The research was focused solely on discrete deterministic uniform synchronous elementary cellular automata with the rectangular shape of cells that are organized into one-dimensional (1D) or two-dimensional (2D) lattice. Moreover, the simplest form of cellular neighborhoods will be considered (i.e. the 3-cell neighborhood in 1D and 5-cell von Neumann's or 9-cell Moore's neighborhood in 2D). The development of these automata is performed in discrete time steps from a given initial configuration where all cells update their states at the same time (synchronously) according to a single transition function (the notion of uniformity) which calculates the new cell states only from their current states (the notion of elementary CA). This concept allows preserving all advantageous features assumed in CA, in particular the simplicity of cells and transition rules, the locality of cellular interactions and scalability of the lattice.

The objective was to investigate advanced representations of the CA transition functions in order to make their evolutionary synthesis more efficient even for multi-state CA. At the beginning, some initial experiments were conducted considering the evolution of transition rules for self-replicating structures in binary 2D CA. However, this showed to be a challenging task if the basic table-based representation was utilized. Therefore, advanced representations have been proposed: (1) an instruction-based approach and (2) Conditionally Matching Rules. The overview of their research is presented in the next sections.

4.1 Instruction-Based Approach

In order to overcome the difficulties related to the CA evolution using the table-based representation, a method for encoding the transition functions by means of an instruction-based approach was proposed the details of which are presented in Appendix I.

This method represents an indirect mapping between the input combinations of states in the cellular neighborhood and the next states of the cells during the CA development. In this case the local transition function is described by a *program* (algorithm) whose execution calculates the next cell states. The objective of the program-based representation is to reduce the length of the chromosomes needed to encode the transition rules the synthesis of which ought to be performed by means of Genetic Algorithm.

It was shown that by using the instruction-based approach the transition function for a given problem may be designed in substantially shorter time and with higher success rate in comparison with the conventional (table-based) approach (especially for multi-state CA). The case studies include the replication problem and the problem of development of a given pattern from an initial seed.

4.2 Introduction of Conditionally Matching Rules

Subsequent experiments with the program-based encoding, however, exhibited difficulties when the number of cell states increased. Moreover, if the program-based approach is applied, then the next state is represented by a result of execution of a prescription (the program) with a limited possibility to detect particular states of the cellular neighborhood for which specific or separate rules could be more suitable. Further analysis of the results presented in Appendix I also showed that the program-based approach is somehow limited in the form of the behavior the CA is finally able to perform. These issues lead to a continuation of the research with the focus on further experimentation with the rule encoding. This resulted in a new rule-based representation which was denominated as *Conditionally Matching Rules* (CMRs). This way a transition function is composed of a finite sequence of CMRs the number of which may be specified by the designer as a parameter. Therefore it allows reducing the length of the chromosomes while preserving a possibility to express individual transition rules as in the case of the table-based encoding.

The proposed CMR approach, described in more detail in Appendix II, is based on conditions specified within the transition rules that have to be satisfied with respect to the current combination of states in the cellular neighborhood in order to determine the next state of a cell according to a specific CMR. The goal of this representation was to be more efficient than the aforementioned table- or program-based approach, yet allowing us to describe specific transition rules in a way naturally convenient to cellular automata. The first experiments were conducted with the replication problem and pattern transformation problem in 2D CA. The initial results presented in Appendix II showed that the evolution is able to design transition functions even for non-trivial behavior of CA that perfectly fulfilled the specified requirements.

The concept of Conditionally Matching Rules represents the main outcome of the research presented in this work. As it showed promising in the initial study, the subsequent research was focused just on this concept. The following sections describe advanced CA benchmarks and applications of CMRs.

4.3 Evolving Computations in Binary 2D CA

The continuation of the CMR research lead to experiments the goal of which was to verify the abilities of this approach in combination with Genetic Algorithms to design given computational process in the 2D cellular array. A detailed description is presented in Appendix III. The main idea is to interpret some predetermined cells as input operands and some (possibly other) cells as output bits (i.e. the result of the computation). The genetic algorithm was applied to find a suitable transition functions for CA according to which the given computation could be observed during its development for all the possible binary combinations stored in the input cells. Both the input values and the result is represented by state values of the given cells. The input values are represented by the initial state of

the CA. After a finite number of development steps the cells representing the output bits are expected to contain the result of the computation. A set of experiments were conducted considering various setups of the evolutionary system and arrangements of the target computation in the CA. It was shown that such computations can be realized in a uniform two-dimensional cellular array. This study also presents a comparative evaluation of the evolutionary experiments for various given scenarios in order to determine abilities of the system to solve this kind of problems.

Note that the research from Appendix III represents the only scenario in which solely binary CA with Moore's neighborhood were considered. It also allowed us to verify the applicability of the CMRs in CA working with larger (9-cell) neighborhoods. All subsequent experiments, introduced in the next sections, will consider multi-state CA (working with at least 6 cell states) with von Neumann's neighborhood.

4.4 Evolving Generic Computation in Multi-State CA

The main study regarding the Conditionally Matching Rules is presented in Appendix IV. Two main areas have been investigated: (1) the evolution of generic square calculations in 1D CA which will be briefly introduced in this section and (2) the design of replicating loop-based structures in 2D CA – this will be mentioned in Section 4.5.

The problem of generic square calculations was inspired by the work of Wolfram [127] where the transition rules performing this computational process in the CA were designed manually. The main idea was to encode the input value $n > 0$ in the initial CA configuration and the result is obtained by performing a finite number of CA steps until the configuration stabilizes. This final configuration encodes (in a given manner) the result of the computation, n^2.

The results presented in Section III of Appendix IV show that it is possible to automatically evolve transition rules (in the CMR form) for calculating the square for various $n > 0$. Moreover, subsequent analysis showed that the resulting CA exhibit substantially lower number of steps needed to calculate the results in comparison with the solution from [127].

A continuation of this research was published by the author of this thesis in [20] and its extended version is included as Appendix VI. Section 4 of Appendix VI proposes a wider analysis of the resulting square calculating CA. In particular, it is demonstrated that various generic CA-based solutions of the squaring problem can be discovered, which substantially overcome the existing solutions regarding both the complexity of the transition functions and the number of steps (speed) of the calculation.

4.5 Evolution of Replicating Structures

In additional to the work of von Neumann [89] with his theoretical study of self-replication in CA (whose solution nevertheless exhibited a significant degree of complexity and has never been physically implemented), several other researchers proposed a concept of so-called self-replicating loops – specific small structures in multi-state CA together with their appropriate (usually manually designed) transition rules for the replication of these structures (see mainly [70, 26, 97]). These structures provided the main inspiration for the research presented in Section IV od Appendix IV.

In this work similar loop-based structures were investigated where the rules for their replication were generated by means of EA. It was shown that more efficient results regarding the replication speed and the complexity of the resulting CA may be discovered using the CMR-based representation in comparison with the existing solutions of self-replicating loops. In particular a novel *diagonal* replication scheme (discovered by the evolution) was presented that exhibits a remarkably high replication speed in comparison with the known solutions.

Further studies of replicating structures were published by the author of this thesis in [19, 18]. An extended version of [19] is presented in Appendix V. It was shown that new replicating loops can be discovered that exhibit some unconventional features in comparison with the known solutions. In particular, several scenarios were presented which can, in addition to the replication from the initial loop, autonomously develop the given loop from a seed (a single "active" cell), with the ability of such loop to subsequently produce its own replicas. Moreover, a *parallel* replicator was presented that is able to develop the replicas to several directions using different replication algorithms.

4.6 Pattern Formation in CA Using CMRs

Advanced experiments with the evolution of CMR-based CA were conducted dealing with the non-trivial pattern formation problem from a seed in two-dimensional CA. Some initial results are presented in Section 5 of Appendix VI. The objective of this part of research was to verify the potential of CMRs to evolve CA producing some given exact stable patterns. The incorporated CA work with 10 and 12 cell states which induce search spaces of enormous sizes. Despite a low success rates of the initial evolutionary experiments some successful results were obtained demonstrating that the automatic design of such CA can be realized. These results probably represent the first case of the automatic evolution of exact behavior in CA with more than 10 cell states.

4.7 Impact of the Rule Encoding on the CA Behavior

Recent experiments with CMRs and EAs lead to a more systematic comparison of this representation with the evolution of CA by means of conventional table-based encoding. The comparison of these results was enabled by applying a custom variant of GA which provided working solutions for both representations of the transition function.

The main results are presented in Appendix VII and a more detailed evaluation was proposed in [21]. The French flag development from a seed was considered as a basic case study. The results showed some remarkable differences in the cellular automata behavior which are probably caused by the utilization of the two distinct representations. Particularly it was determined that CMRs allow the CA to develop the pattern gradually from the seed and even to achieve a stable final configuration. On the other hand the table representation always exhibited an emergence of a chaotic state from which the pattern eventually appeared but was destroyed during subsequent CA development and never restored again.

The obtained results demonstrated a necessity of further research of the CA rule encoding in general. So far the CMRs showed some aspects of the CA development which have never been observed before (in particular the issues of gradual seed growth in multi-state CA with a pattern stability or possibility to obtain a given exact CA behavior with more than 10 cell states).

4.8 Application of CMRs on Image Filtering

In order to present the concept of CMRs in CA applied in a real-world area, the task of noisy image filtering was chosen as a case study. In Appendix VIII an evolutionary method for the design of image filters using 2D CA is presented. In particular the filtering of Salt-and-Pepper noise from 8-bit gray-scale images is considered. The CMRs are applied to design CA whose cells represent pixels of the image. In case of the 8-bit images the CA works with 256 cell states. The utilization of CMRs for representing and evolving the filtering functions is actually a need in this application since the encoding using the traditional table-based method seems to be inapplicable for such a high number of states.

The main ideas of the proposed approach are the following. A cellular automaton was initialized by the values of pixels of a corrupted image and a variant of Evolution Strategy was applied for the design of a suitable transition function able to eliminate the noise from the image by means of ordinary development of the cellular automaton. It was shown that using only 5-cell neighborhood of the CA in combination with CMRs the resulting filters were able to provide a very good output quality and are comparable with several existing solutions that required more resources. Moreover, the proposed evolutionary method exhibited a high performance which allows us to design filters in very short time even on a common PC.

Chapter 5

Conclusion

The research presented in this work investigated new representations of transition functions for the evolutionary design of cellular automata. The goal was to enable the evolutionary algorithms to successfully design complex multi-state cellular automata which have not been possible to obtain before by means of conventional table-based representation. Several case studies have been considered including computations in the cellular array, replication problem, pattern development problem or image denoising. This chapter summarizes the obtained results and discusses some possibilities of future work.

5.1 Contributions

Two new representations have been proposed which have not been applied in the CA design so far: the instruction-based representation and Conditionally Matching Rules. The latter approach showed to be promising for future research. Whilst the transition functions represented as programs were inspired by techniques like Genetic Programming or similar concepts, where "callable" functions may be defined as more complex blocks for controlling a system, they showed rather a limited applicability for the CA evolution. On the other hand, the CMRs are closer to the original CA control in the form of separate rules according to which the individual cells may develop. Finally (in this research) this concept showed to be suitable for a variety of tasks that ought to be solved by cellular automata.

The results in all the considered case studies showed an evident impact the the CMR representation on both the evolutionary design process of the CA (a higher success rate in most sets of experiments) and the behavior of the CA themselves. In addition, the CMRs allowed us to design behavior in multi-state CA which has not been achieved before by means of the conventional representation (the replicating loops or development of stable patterns from a seed represent the most remarkable examples). Therefore it may be concluded that **the hypothesis stated on page 7 was confirmed by these experiments**. The concept of Conditionally Matching Rules represents the main scientific contribution of this thesis.

5.2 Main Outcomes

In addition to the proposed representation techniques that allowed the evolution of non-trivial multi-state cellular automata, the results of various case studies which constituted the main issues of interest in this work represent the particular outcomes of this research. Let us

briefly summarize the most interesting CA obtained from the experiments and observations related to these results.

The instruction-based approach provided CA with the behavior that exactly matches the initial specification. In particular, replication of patterns of various shapes and sizes were successfully realized. The second successful benchmark was a simple pattern development from a seed. In both cases binary as well as multi-state cellular automata were considered. **It may be stated the first goal from page 8 was fulfilled by this part of research.**

The introduction of Conditionally Matching Rules allowed the CA to perform significantly more complex behavior discovered by the evolution for various problems. Although the initial experiments with CMRs were performed on binary CA, the proposed concept proved its advantages in various applications, specifically the replication problem, pattern transformation problem and calculations in 2D cellular array. In all these cases the CMRs allowed obtaining advanced CA whose behavior was difficult to obtain by means of other known methods (or even such experiments were unsuccessful so far). **The invention of a new innovative representation for the evolution of CA can be considered as an achievement of goal 2 from page 8**

The real demonstration of the abilities of Conditionally Matching Rules was performed during subsequent research where advanced benchmark problems solven solely by means of multi-state CA were investigates. In particular, various processes for generic square calculations in 1D CA were successfully evolved where some of these solutions outperformed the existing method. Moreover, the problem of designing replicating loops in 2D CA was considered the results of which provided some interesting concepts allowing to generate more copies of the given structure in comparison with similar known solutions. The CMRs also succeeded in advanced pattern development problem from a seed where even stable structures exhibiting a movement across the cellular array were sometimes obtained as a side effect that was not the part of the original specification. **The successful solution of various advanced benchmarks by means of CMRs was the fulfillment of goal 3 from page 8**

A more in-depth study was performed in order to evaluate the specific type of behavior of CA driven by Conditionally Matching Rules in comparison with the conventional table-based representation. The pattern development from a seed was considered as a case study. The experiments showed some interesting observations. In case of the utilization of CMRs the CA behavior exhibited a gradual "growth" of the seed which eventually finished in a given (stable) target pattern. However, when the conventional encoding was applied, the seed developed into a "random" (chaotic) global state from which the target pattern emerged after a while literally within a single step from the chaos. However, such patterns were always destroyed by subsequent CA development and have never restored again (i.e. the conventional representation failed in the evolution of stable solution). Moreover, some behavior was not possible to obtain at all by means of the table-based approach (e.g. the replication of the loops). **The demonstration of the impact of CMRs on the CA behavior accomplished goal 4 from page 8.**

Finally, the image denoising task was considered as an application inspired by real-world problems. The CMRs were applied to design filtering functions of 8-bit gray-scale images (i.e. the CA worked with 256 states representing the pixel shades) corrupted by Salt-and-Pepper noise. The initial design (training) of the filters were performed with 10% noise, the testing was performed with various images corrupted by the noise whose intensity achieved up to 90%. The proposed method allowed designing good filters within quite short time periods provided to the evolution (e.g. less than 10 minutes). The best obtained filters were

qualitatively comparable with some existing solutions. **This part of work has fulfilled the last goal stated on page 8**

5.3 Future Work

The issue related to the encoding of candidate solutions for EA represents one of the key issue in evolutionary design and optimization in general. The proposed work demonstrated that this is no less important for the evolution of CA. It is evident from the results obtained by means of CMRs that the representation may have a significant impact on the resulting CA behavior. Hence this method seems promising for future research.

So far, only basic CA variants have been considered. Nevertheless this does not mean that their design is trivial. Here "basic" means rather the CA with smallest cellular neighborhood (i.e. 3-cell neighborhood in 1D CA and 5 cells for 2D lattice). However, even those scenarios induce huge search spaces for multi-state CA which grows exponentially with enhancing the neighborhood size. One of the issues for future work is to investigate the abilities of the new representations to provide solutions for CA with such enhancements which is common in cases of performing complex simulations by means of CA (e.g. simulation of physical systems, transportation, crowd behavior etc.).

The proposed concepts (the program-based representation and Conditionally Matching Rules) were tuned so far on some (mostly benchmark) problems for which a suitable configurations were found (i.e. the instruction set for the programs and the set of conditions for CMRs). However, both approaches are quite robust which allows them to be "configured" in many ways. The issue of determining suitable configurations for given tasks in CA (the target CA behavior) still remain an open question for which no systematic approach exist and further research would be needed to apply these methods in other areas.

Finally, some of the most complex setups of the evolutionary CA design are those where the CA constitute hybrid systems. In this context, as a hybrid system may be, for example, considered a CA whose transition function is calculated by a neural network or an other (usually well-established) model. Similarly the transition function may be formed by combining several (basic) approaches, e.g. the next cell state may be determined by a program for one condition identified in the cellular neighborhood or by a transition table for another condition. This setup forms a hybrid system combining the CA and several different ways controlling its development. This issue may be studied during future advanced research.

Bibliography

[1] Abdennour, I., Ouardouz, M. and Bernoussi, A. S. Modeling of Electrical and Thermal Behaviors of Photovoltaic Panels Using Cellular Automata Approach. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115.* Cham: Springer International Publishing, 2018, p. 57–67.

[2] Adak, S., Mukherjee, S. and Das, S. Do There Exist Non-linear Maximal Length Cellular Automata? A Study. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115.* Cham: Springer International Publishing, 2018, p. 289–297.

[3] Adamatzky, A. *Identification of Cellular Automata.* CRC Press, 1994.

[4] Adamatzky, A. *Game of Life Cellular Automata.* 1st ed. Springer Publishing Company, Incorporated, 2010. ISBN 1849962162, 9781849962162.

[5] Adamatzky, A. Identification of Cellular Automata. In: Meyers, R. A., ed. *Computational Complexity.* Springer New York, 2012, p. 1564–1575.

[6] Alfonseca, M. and Gil, F. Evolving Interesting Initial Conditions for Cellular Automata of the Game of Life Type. *Complex Systems.* march 2012, vol. 21, p. 57–70.

[7] Alonso Sanz, R. Cellular automata and other discrete dynamical systems with memory. In: *2012 International Conference on High Performance Computing & Simulation (HPCS).* IEEE, 2012, p. 215–215.

[8] Andre, D., Bennett, F. H. and Koza, J. R. Discovery by Genetic Programming of a Cellular Automata Rule That is Better Than Any Known Rule for the Majority Classification Problem. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation.* MIT Press, 1996, p. 3–11.

[9] Bagnoli, F., Dridi, S., Yacoubi, S. E. and Rechtman, R. Regional Control of Probabilistic Cellular Automata. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115.* Cham: Springer International Publishing, 2018, p. 243–254.

[10] Bagnoli, F. and Rechtman, R. Regional Synchronization of a Probabilistic Cellular Automaton. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari,

K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115.* Cham: Springer International Publishing, 2018, p. 255–263.

[11] Bagnoli, F., Yacoubi, S. E. and Rechtman, R. Regional Control of Boolean Cellular Automata. In: El Yacoubi, S., Wąs, J. and Bandini, S., ed. *Cellular Automata ACRI 2016. Lecture Notes in Computer Science, vol 9863.* Cham: Springer International Publishing, 2016, p. 101–112.

[12] Bakhteri, R., Cheng, J. and Semmelhack, A. Design and Implementation of Cellular Automata on FPGA for Hardware Acceleration. *Procedia Computer Science.* 2020, vol. 171, p. 1999–2007. ISSN 1877-0509. Third International Conference on Computing and Network Communications (CoCoNet'19).

[13] Bandini, S., Crociani, L. and Vizzari, G. An Approach for Managing Heterogeneous Speed Profiles in Cellular Automata Pedestrian Models. *Journal of cellular automata.* 2017, vol. 12, no. 5, p. 401–421.

[14] Barik, R., Naskar, M. N. B. J., Modak, S. and Verma, U. Analysis and Application of Cellular Automata Based Edge Detection Methods on Bio-Medical Images. *International Journal of Engineering Research & Technology (IJERT).* 2021, vol. 9, no. 11, p. 143–148. ISSN 1877-0509.

[15] Barman, D. and Das, S. A Cellular Automaton that Computes Shortest Paths in Grid Graph. In: Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaski, K., ed. *Cellular Automata ACRI 2020. Lecture Notes in Computer Science, vol 12599.* Cham: Springer International Publishing, 2021, p. 3–7.

[16] Ben Jacob, E., Schochet, O., Tenenbaum, A., Cohen, I., Czirók, A. et al. Generic modelling of cooperative growth patterns in bacterial colonies. *Nature.* 2011, vol. 368, p. 46–49.

[17] Bhaumik, J., Roy Chowdhury, D. and Chakrabarti, I. Null Boundary 90/150 Cellular Automata for Multi-byte Error Correcting Code. In: Bandini, S., Manzoni, S., Umeo, H. and Vizzari, G., ed. *Cellular Automata ACRI 2010. Lecture Notes in Computer Science, vol 6350.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, p. 231–240.

[18] Bidlo, M. Investigation of Replicating Tiles in Cellular Automata Designed by Evolution Using Conditionally Matching Rules. In: *2015 IEEE International Conference on Evolvable Systems (ICES).* IEEE Computational Intelligence Society, 2015, p. 1506–1513. Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence (SSCI).

[19] Bidlo, M. On Routine Evolution of New Replicating Structures in Cellular Automata. In: *7th International Conference on Evolutionary Computationa Theory and Applications.* SciTePress - Science and Technology Publications, 2015, p. 28–38. 7th International Joint Conference on Computational Intelligence.

[20] Bidlo, M. Evolution of Generic Square Calculations in Cellular Automata. In: *Proceedings of the 8th International Joint Conference on Computational Intelligence - Volume 3: ECTA.* SciTePress - Science and Technology Publications, 2016, p. 94–102. ISBN 978-989-758-201-1.

[21] Bidlo, M. Comparison of Evolutionary Development of Cellular Automata Using Various Representations. *MENDEL Soft Computing Journal*. 2019, vol. 2019, no. 1, p. 95–102.

[22] Bołt, W., Baetens, J. M. and De Baets, B. An evolutionary approach to the identification of Cellular Automata based on partial observations. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2015, p. 2966–2972. DOI: 10.1109/CEC.2015.7257258.

[23] Bołt, W., Bołt, A., Wolnik, B., Baetens, J. M. and De Baets, B. A statistical approach to the identification of diploid cellular automata based on incomplete observations. *Biosystems*. 2019, vol. 186, p. 103976. ISSN 0303-2647.

[24] Bull, L., Lawson, I., Adamatzky, A. and DeLacyCostello, B. Towards Predicting Spatial Complexity: A Learning Classifier System Approach to the Identification of Cellular Automata. In: *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*. IEEE Computer Society, 2005, p. 136–141.

[25] Byari, M., Bernoussi, A., Amharref, M. and Ouardouz, M. A 3D Cellular Automata Approach for the Wind Flow Modeling. *Journal of Cellular Automata*. 2022, vol. 16, 3–4, p. 213–237.

[26] Byl, J. Self-reproduction in small cellular automata. *Physica D: Nonlinear Phenomena*. 1989, vol. 34, 1–2, p. 295–299.

[27] Cagigas Muñiz, D., Rio, F. Diaz-del, López Torres, M., Jiménez Morales, F. and Guisado, J. L. Developing Efficient Discrete Simulations on Multicore and GPU Architectures. *Electronics*. january 2020, vol. 9, p. 189.

[28] Cagigas Muñiz, D., Rio, F. D. del, Sevillano Ramos, J. L. and Guisado Lizar, J.-L. Efficient simulation execution of cellular automata on GPU. *Simulation Modelling Practice and Theory*. 2022, vol. 118. ISSN 1569-190X.

[29] Cenek, M. and Mitchell, M. Evolving Cellular Automata. In: Meyers, R. A., ed. *Computational Complexity: Theory, Techniques, and Applications*. New York, NY: Springer New York, 2012, p. 1043–1052.

[30] Chakraborty, S., Kundu, S. and Chowdhury, D. R. Image Encryption with Parallel Evolution of 2-D Cellular Automata. In: Giri, D., Raymond Choo, K.-K., Ponnusamy, S., Meng, W., Akleylek, S. et al., ed. *Proceedings of the Seventh International Conference on Mathematics and Computing*. Singapore: Springer Singapore, 2022, p. 63–78.

[31] Chowdhury, D., Gupta, I. and Chaudhuri, P. CA-based byte error-correcting code. *IEEE Transactions on Computers*. 1995, vol. 44, no. 3, p. 371–382.

[32] Cook, M. Universality in Elementary Cellular Automata. *Complex Systems*. Complex Systems Publications, Inc. 2004, vol. 15, no. 1, p. 1–40.

[33] Das, A. K. and Chattaraj, U. Cellular Automata Model for Lane Changing Activity. *International Journal of Intelligent Transportation Systems Research*. 2022, vol. 20, p. 446–455. ISSN 1868-8659.

[34] Das, S. and Roy Chowdhury, D. CASTREAM: A New Stream Cipher Suitable for Both Hardware and Software. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2012. Lecture Notes in Computer Science, vol 7495.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 601–610.

[35] Dennunzio, A., Formenti, E. and Kůrka, P. Cellular Automata Dynamical Systems. In: Rozenberg, G., Bäck, T. and Kok, J. N., ed. *Handbook of Natural Computing.* Springer Berlin Heidelberg, 2012, p. 25–75.

[36] Dewdney, A. K. Computer Recreations. *Scientific American.* Scientific American, a division of Nature America, Inc. 1990, vol. 262, no. 1, p. 146–149.

[37] Elmenreich, W. and Fehérvári, I. Evolving Self-organizing Cellular Automata Based on Neural Network Genotypes. In: *Proc. of the 5th International Conference on Self-organizing Systems.* Springer, 2011, p. 16–25.

[38] Enescu, A., Dumitru, D., Andreica, A. and Dioşan, L. Unsupervised Edge Detector based on Evolved Cellular Automata. *Procedia Computer Science.* 2020, vol. 176, p. 470–479. ISSN 1877-0509. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

[39] Farner, J. J., Weydahl, H., Jahren, R., Ramstad, O. H., Nichele, S. et al. Evolving spiking neuron cellular automata and networks to emulate in vitro neuronal activity. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI).* IEEE, 2021.

[40] Fatès, N. A Guided Tour of Asynchronous Cellular Automata. *Journal of Cellular Automata.* 2014, vol. 9, 5-6, p. 387–416.

[41] Fatès, N. Asynchronous Cellular Automata. In: Adamatzky, A., ed. *Cellular Automata: A Volume in the Encyclopedia of Complexity and Systems Science, Second Edition.* New York, NY: Springer US, 2018, p. 73–92.

[42] Formenti, E. and Paquelin, J.-L. High Order Cellular Automata for Edge Detection: A Preliminary Study. In: Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaski, K., ed. *Cellular Automata ACRI 2020. Lecture Notes in Computer Science, vol 12599.* Cham: Springer International Publishing, 2021, p. 80–89.

[43] Fraga, L. M., Oliveira, G. M. B. de and Martins, L. G. A. Multistage Evolutionary Strategies for Adjusting a Cellular Automata-based Epidemiological Model. In: *2021 IEEE Congress on Evolutionary Computation (CEC).* IEEE, 2021, p. 466–473.

[44] Gadouleau, M., Mariot, L. and Picek, S. *Bent Functions from Cellular Automata* [Cryptology ePrint Archive, Paper 2020/1272]. 2020.

[45] Gardner, M. Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game "Life". *Scientific American.* 1970, vol. 223, October 1970, p. 120–123.

[46] Gerakakis, I., Gavriilidis, P., Dourvas, N. I., Georgoudas, I. G., Trunfio, G. A. et al. Accelerating fuzzy cellular automata for modeling crowd dynamics. *Journal of Computational Science*. 2019, vol. 32, p. 125–140. ISSN 1877-7503.

[47] Ghosh, S., Sengupta, A., Saha, D. and Chowdhury, D. R. A Scalable Method for Constructing Non-linear Cellular Automata with Period $2^n - 1$. In: Wąs, J., Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2014. Lecture Notes in Computer Science, vol 8751*. Cham: Springer International Publishing, 2014, p. 65–74.

[48] Girau, B. and Vlassopoulos, N. Evolution of 2-Dimensional Cellular Automata as Pseudo-random Number Generators. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2012. Lecture Notes in Computer Science, vol 7495*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 611–622.

[49] Gorrini, A., Crociani, L., Vizzari, G. and Bandini, S. Cumulative Mean Crowding and Pedestrian Crowds: A Cellular Automata Model. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115*. Cham: Springer International Publishing, 2018, p. 481–491.

[50] Guidolin, M., Chen, A. S., Ghimire, B., Keedwell, E. C., Djordjevic, S. et al. A weighted cellular automata 2D inundation model for rapid flood analysis. *Environmental Modeling and Software*. 2016, vol. 84, p. 378–394.

[51] Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaskil, K. *(eds.) 14th International Conference on Cellular Automata for Research and Industry, ACRI 2020, Lodz, Poland, December 2–4, 2020, Proceedings, LNCS vol. 12599*. Cham: Springer, 2021.

[52] Hassan, Y. Fractional integral model in cellular automata embedded for biometric recognition. *ICIC Express Letters*. june 2018, vol. 12, p. 607–614.

[53] Hazari, R., Kundu, S., Bhardwaj, M. and Das, S. ECA 184 can implement any logic circuits. *Journal of Cellular Automata*. january 2018, vol. 13, p. 359–371.

[54] Heaton, J. Evolving Continuous Cellular Automata for Aesthetic Objectives. *Genetic Programming and Evolvable Machines*. USA: Kluwer Academic Publishers. 2019, vol. 20, no. 1, p. 93–125. ISSN 1389-2576.

[55] Hedlund, G. A. Compact CA-Based Single Byte Error Correcting Codec. *Mathematical systems theory*. 1969, vol. 3, p. 320–375. ISSN 1432-4350.

[56] Hess, K. Nano-Structures, Quantum Computing and Cellular Automata. *Journal of Computational and Theoretical Nanoscience*. 2011, vol. 8, no. 6.

[57] Ilachinski, A. *Cellular Automata: A Discrete Universe*. World Scientific, 2001.

[58] Ishibuchi, H., Nozaki, K. and Yamamoto, N. Selecting fuzzy rules by genetic algorithm for classification problems. In: *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. 1993, p. 1119–1124 vol.2.

[59] Ishida, T. and Inokuchi, S. Limit Cycle for Composited Cellar Automata. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2012. Lecture Notes in Computer Science, vol 7495.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 32–41.

[60] Isokawa, T., Peper, F., Ono, K. and Matsui, N. A universal Brownian cellular automaton with 3 states and 2 rules. *Natural Computing.* 2018, vol. 17, p. 499–509. ISSN 1572-9796.

[61] Jellouli, O., Bernoussi, A., Amharref, M. and Ouardouz, M. Modeling of Wind Flow and Its Impact on Forest Fire Spread: Cellular Automata Approach. In: El Yacoubi, S., Wąs, J. and Bandini, S., ed. *Cellular Automata ACRI 2016. Lecture Notes in Computer Science, vol 9863.* Cham: Springer International Publishing, 2016, p. 269–279.

[62] John, A., Nandu, B. C., Ajesh, A. and Jose, J. PENTAVIUM: Potent Trivium-Like Stream Cipher Using Higher Radii Cellular Automata. In: Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaski, K., ed. *Cellular Automata ACRI 2020. Lecture Notes in Computer Science, vol 12599.* Cham: Springer International Publishing, 2021, p. 90–100.

[63] Jose, J., Das, S. and Chowdhury, D. R. Inapplicability of Fault Attacks against Trivium on a Cellular Automata Based Stream Cipher. In: Wąs, J., Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2014. Lecture Notes in Computer Science, vol 8751.* Cham: Springer International Publishing, 2014, p. 427–436.

[64] Kamilya, S., Das, S. and Sikdar, B. K. Simulation of Non-uniform Cellular Automata by Classical Cellular Automata and Its Application in Embedded Systems. *Journal of Cellular Automata.* 2021, vol. 16, 1–2, p. 61–86.

[65] Korček, P., Sekanina, L. and Fučík, O. Advanced Approach to Calibration of Traffic Microsimulation Using Travel Times. *Journal of Cellular Automata.* Old City Publishing, Inc. 2013, vol. 8, no. 6, p. 457–467.

[66] Kutrib, M. and Malcher, A. One-dimensional pattern generation by cellular automata. *Natural Computing.* 2022, vol. 2022. ISSN 1572-9796.

[67] Kyparissas, N. and Dollas, A. Large-Scale Cellular Automata on FPGAs: A New Generic Architecture and a Framework. *ACM Trans. Reconfigurable Technol. Syst.* New York, NY, USA: Association for Computing Machinery. 2020, vol. 14, no. 1. ISSN 1936-7406.

[68] Lakra, R., John, A. and Jose, J. CARPenter: A Cellular Automata Based Resilient Pentavalent Stream Cipher. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K. and Manzoni, L., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 11115.* Cham: Springer International Publishing, 2018, p. 352–363.

[69] Land, M. and Belew, R. K. No Perfect Two-State Cellular Automata for Density Classification Exists. *Physical Review Letters.* American Physical Society. 1995, vol. 74, p. 5148–5150.

[70] Langton, C. G. Self-Reproduction in Cellular Automata. *Physica D: Nonlinear Phenomena*. 1984, vol. 10, 1–2, p. 135–144.

[71] Langton, C. G. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena*. 1986, vol. 22, 1–3, p. 120–149.

[72] Liu, Y., Batty, M., Wang, S. and Corcoran, J. Modelling urban change with cellular automata: Contemporary issues and future research directions. *Progress in Human Geography*. 2021, vol. 45, no. 1.

[73] Lohn, J. and Reggia, J. Discovery of self-replicating structures using a genetic algorithm. In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. 1995, vol. 2, p. 678–683 vol.2.

[74] Lohn, J. and Reggia, J. Automatic discovery of self-replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation*. 1997, vol. 1, no. 3, p. 165–178.

[75] Maeda, K.-I. and Sakama, C. Identifying Cellular Automata Rules. *Journal of Cellular Automata*. Old City Publishing. 2007, vol. 2, no. 1, p. 1–20.

[76] Mariot, L., Picek, S., Jakobovic, D. and Leporati, A. Evolutionary algorithms for designing reversible cellular automata. *Genetic Programming and Evolvable Machines*. Kluwer Academic Publishers. 2021, vol. 2021, p. 429–461.

[77] Mariot, L., Saletta, M., Leporati, A. and Manzoni, L. Exploring Semi-bent Boolean Functions Arising from Cellular Automata. In: Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaski, K., ed. *Cellular Automata ACRI 2020. Lecture Notes in Computer Science, vol 12599*. Cham: Springer International Publishing, 2021, p. 56–66.

[78] Mauri, G., Yacoubi, S. E., Dennunzio, A., Nishinari, K. and Manzoni, L. *(eds.) 13th International Conference on Cellular Automata for Research and Industry, ACRI 2018, Como, Italy, September 17–21, 2018, Proceedings, LNCS vol. 11115*. Cham: Springer, 2018.

[79] Mitchell, M. Computation in Cellular Automata: A Selected Review. In: *Non-Standard Computation*. John Wiley & Sons, 1998, chap. 4, p. 95–140. ISBN 9783527602964.

[80] Mitchell, M., Crutchfield, J. and Das, R. Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work. *First Int. Conf. on Evolutionary Computation and Its Applications*. may 1996, vol. 1.

[81] Mitsopoulou, M., Dourvas, N., Georgoudas, I. G. and Sirakoulis, G. C. Cellular Automata Model for Crowd Behavior Management in Airports. In: Wyrzykowski, R., Deelman, E., Dongarra, J. and Karczewski, K., ed. *Parallel Processing and Applied Mathematics PPAM 2019. Lecture Notes in Computer Science, vol 12044*. Cham: Springer International Publishing, 2020, p. 445–456.

[82] Mitsopoulou, M., Dourvas, N. I., Sirakoulis, G. C. and Nishinari, K. Spatial games and memory effects on crowd evacuation behavior with Cellular Automata. *Journal of Computational Science*. 2019, vol. 32, p. 87–98. ISSN 1877-7503.

[83] Mordvintsev, A., Randazzo, E. and Fouts, C. *Growing Isotropic Neural Cellular Automata*. arXiv, 2022. DOI: 10.48550/ARXIV.2205.01681. Available at: https://arxiv.org/abs/2205.01681.

[84] Mordvintsev, A., Randazzo, E., Niklasson, E. and Levin, M. Growing Neural Cellular Automata. *Distill*. 2020. DOI: 10.23915/distill.00023. https://distill.pub/2020/growing-ca. Available at: https://distill.pub/2020/growing-ca/.

[85] Morita, K. Universality of 8-State Reversible and Conservative Triangular Partition ed Cellular Automata. In: El Yacoubi, S., Wąs, J. and Bandini, S., ed. *Cellular Automata ACRI 2018. Lecture Notes in Computer Science, vol 9863*. Cham: Springer International Publishing, 2016, p. 45–54.

[86] Naskar, N., Adak, S., Maji, P. and Das, S. Synthesis of Non-uniform Cellular Automata Having only Point Attractors. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2014. Lecture Notes in Computer Science, vol 8751*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, p. 105–114.

[87] Nehaniv, C. Asynchronous Automata Networks Can Emulate any Synchronous Automata Network. *International Journal of Algebra and Computation (IJAC)*. october 2004, vol. 14, p. 719–739.

[88] Nehaniv, C. L. Evolution in Asynchronous Cellular Automata. In: *Proceedings of the Eighth International Conference on Artificial Life*. Cambridge, MA, USA: MIT Press, 2002, p. 65–73. ICAL 2003.

[89] Neumann, J. von. *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[90] Neumann, J. von and Kurzweil, R. *The Computer and the Brain, 3rd ed.* Yale University Press, 2012.

[91] Ninagawa, S. and Martínez, G. J. Power Spectral Analysis of the Computation Process by Rule 110. In: Wąs, J., Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2014. Lecture Notes in Computer Science, vol 8751*. Cham: Springer International Publishing, 2014, p. 45–54.

[92] Ohi, F. and Ichikawa, T. Confliction-Like Dynamics of Rule 20 ECA of Wolfram Class II. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata ACRI 2012. Lecture Notes in Computer Science, vol 7495*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 73–82.

[93] Pan, Z. and Reggia, J. Evolutionary Discovery of Arbitrary Self-replicating Structures. In: Sunderam, V. S., Albada, G. D. van, Sloot, P. M. A. and Dongarra, J. J., ed. *Computational Science – ICCS 2005*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, p. 404–411.

[94] Pan, Z. and Reggia, J. A. Artificial Evolution of Arbitrary Self-Replicating Structures in Cellular Spaces. In: Kroc, J., Sloot, P. M. and Hoekstra, A. G., ed. *Simulating Complex Systems by Cellular Automata*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, p. 193–216.

[95] Piwonska, A., Seredynski, F. and Szaban, M. Searching Cellular Automata Rules for Solving Two-Dimensional Binary Classification Problem. In: Sirakoulis, G. C. and Bandini, S., ed. *Cellular Automata*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 121–130.

[96] Rallis, K., Moysidis, S. and Karafyllidis, I. G. Implementation of Cellular Automata Using Graphene Nanoribbons with Magnetic Contacts. In: Gwizdałła, T. M., Manzoni, L., Sirakoulis, G. C., Bandini, S. and Podlaski, K., ed. *Cellular Automata ACRI 2020. Lecture Notes in Computer Science, vol 12599*. Cham: Springer International Publishing, 2021, p. 169–176.

[97] Reggia, J. A., Armentrout, S. L., Chou, H.-H. and Peng, Y. Simple Systems That Exhibit Self-Directed Replication. *Science*. 1993, vol. 259, no. 5099, p. 1282–1287.

[98] Rendell, P. A Universal Turing Machine in Conway's Game of Life. In: *2011 International Conference on High Performance Computing and Simulation (HPCS)*. 2011, p. 764–772.

[99] Rendell, P. A Fully Universal Turing Machine in Conway's Game of Life. *Journal of Cellular Automata*. 2013, vol. 9, 1–2, p. 19–358.

[100] Rhodes, A. D. Evolving Order and Chaos: Comparing Particle Swarm Optimization and Genetic Algorithms for Global Coordination of Cellular Automata. In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020.

[101] Richard, G. On the Synchronisation Problem over Cellular Automata. In: Vollmer, H. and Vallée, B., ed. *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, vol. 66, p. 54:1–54:13. LIPIcs.

[102] Rosin, P., Adamatzky, A. and Sun, X. *(eds.) Cellular Automata in Image Processing and Geometry. Emergence, Complexity and Computation series, vol. 10*. Cham: Springer International Publishing Switzerland, 2014.

[103] Rosin, P. and Sun, X. Cellular Automata as a Tool for Image Processing. In: *Emerging Topics in Computer Vision and its Applications*. September 2011, p. 233–251. ISBN 978-981-4340-99-1.

[104] Rosin, P. L. and Sun, X. Cellular Automata as a Tool for Image Processing. In: *Emerging Topics in Computer Vision and its Applications*. World Scientific, p. 233–251.

[105] Roy, S. and Adak, S. Asynchronous Cellular Automata as Randomness Enhancer. In: Das, S. and Martinez, G. J., ed. *Proceedings of First Asian Symposium on Cellular Automata Technology*. Singapore: Springer Nature Singapore, 2022, p. 139–151.

[106] Roy, S. and Das, S. Asynchronous cellular automata that hide some of the configurations during evolution. *International Journal of Modern Physics C*. december 2020, vol. 32.

[107] Ruivo, E. L., Balbi, P. P. and Perrot, K. An asynchronous solution to the synchronisation problem for binary one-dimensional cellular automata. *Physica D: Nonlinear Phenomena*. 2020, vol. 413. ISSN 0167-2789.

[108] Sahin, U., Uguz, S. and Sahin, F. Salt and pepper noise filtering with fuzzy-cellular automata. *Computers & Electrical Engineering*. 2014, vol. 40, no. 1, p. 59–69. ISSN 0045-7906.

[109] Samanta, J., Bhaumik, J. and Barman, S. Compact CA-Based Single Byte Error Correcting Codec. *IEEE Transactions on Computers*. Los Alamitos, CA, USA: IEEE Computer Society. 2018, vol. 67, no. 02, p. 291–298. ISSN 1557-9956.

[110] Sapin, E., Adamatzky, A., Collet, P. and Bull, L. Stochastic automated search methods in cellular automata: the discovery of tens of thousands of glider guns. *Natural Computing*. 2010, vol. 9, no. 3, p. 513–543.

[111] Sapin, E. Gliders and Glider Guns Discovery in Cellular Automata. In: *A. Adamatzky (ed.), Game of Life Cellular Automata*. Springer, 2010, p. 135–165.

[112] Sapin, E., Bailleux, O. and Chabrier, J.-J. Research of Complex Forms in Cellular Automata by Evolutionary Algorithms. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E. and Schoenauer, M., ed. *Artificial Evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 357–367.

[113] Sapin, E. and Bull, L. Searching for Glider Guns in Cellular Automata: Exploring Evolutionary and Other Techniques. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M. and Lutton, E., ed. *Artificial Evolution*. Springer Berlin Heidelberg, 2008, vol. 4926, p. 255–265. Lecture Notes in Computer Science.

[114] Sarkar, A., Mukherjee, A. and Das, S. Reversibility in Asynchronous Cellular Automata. *Complex Systems*. 2012, vol. 21, no. 1, p. 71–84.

[115] Sato, S. and Kanoh, H. Evolutionary design of edge detector using rule-changing Cellular automata. In: *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*. IEEE, 2010, p. 60–65.

[116] Sipper, M. *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, Vol. 1194*. Berlin: Springer, 1997.

[117] Sipper, M., Goeke, M., Mange, D., Stauffer, A., Sanchez, E. et al. The firefly machine: online evolware. In: *Evolutionary Computation, 1997., IEEE International Conference on*. 1997, p. 181–186.

[118] Sirakoulis, G. C. and Bandini, S. *(eds.) 10th International Conference on Cellular Automata for Research and Industry, ACRI 2012, Santorini Island, Greece, September 24-27, 2012, Proceedings, LNCS vol. 7495*. Cham: Springer, 2012.

[119] Soto, J. M. G. and Wuensche, A. Logical Universality from a Minimal Two-Dimensional Glider Gun. *Complex Systems*. 2018, vol. 27, no. 1.

[120] Souza, L. F., Filho, T. M. R. and Moret, M. A. Relating SARS-CoV-2 variants using cellular automata imaging. *Nature Scientific Reports*. 2022, vol. 12.

[121] Suresh, A., Sundaranarayana, D. and Kamaleshwar, T. Iterative Refined Noisy Pixel Restoration (IRNPR) Cellular Automaton Based Image Denoising Methods for Biometric Images. *European Journal of Molecular & Clinical Medicine*. 2020, vol. 7, no. 6, p. 2524–2536. ISSN 2515-8260.

[122] Turney, P. D. Evolution of Autopoiesis and Multicellularity in the Game of Life. *Artificial Life*. june 2021, vol. 27, no. 1, p. 26–43. ISSN 1064-5462.

[123] Žaloudek, L., Sekanina, L. and Šimek, V. GPU Accelerators for Evolvable Cellular Automata. In: *Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns*. Institute of Electrical and Electronics Engineers, 2009, p. 533–537.

[124] Žaloudek, L., Sekanina, L. and Šimek, V. Accelerating Cellular Automata Evolution on Graphics Processing Units. *International Journal on Advances in Software*. 2010, vol. 3, no. 1, p. 294–303. ISSN 1942-2628.

[125] Wimpenny, J. W. and Colasanti, R. A unifying hypothesis for the structure of microbial biofilms based on cellular automaton models. *FEMS Microbiology Ecology*. 1997, vol. 22, no. 1, p. 1–16.

[126] Wolfram, S. *Cellular Automata and Complexity*. Addison Wesley, 1994.

[127] Wolfram, S. *A New Kind of Science*. Champaign IL: Wolfram Media, 2002.

[128] Wąs, J., Sirakoulis, G. C. and Bandini, S. *(eds.) 11th International Conference on Cellular Automata for Research and Industry, ACRI 2014, Krakow, Poland, September 22–25, 2014, Proceedings, LNCS vol. 8751*. Cham: Springer, 2014.

[129] Yacoubi, S. E., Wąs, J. and Bandini, S. *(eds.) 12th International Conference on Cellular Automata for Research and Industry, ACRI 2018, Fez, Morocco, September 5–8, 2016, Proceedings, LNCS vol. 9863*. Cham: Springer, 2016.

[130] Zaitsev, D. Simulating Cellular Automata by Infinite Petri Nets. *Journal of cellular automata*. 2018, vol. 13, 1–2, p. 121–144.

[131] Zeng, J., Qian, Y., Yin, F., Zhu, L. and Xu, D. A multi-value cellular automata model for multi-lane traffic flow under lagrange coordinate. *Computational and Mathematical Organization Theory*. 2022, vol. 28, p. 178–192. ISSN 1572-9346.

[132] Zhao, Y. and Billings, S. A. The Identification of Cellular Automata. *Journal of Cellular Automata*. Old City Publishing. 2007, vol. 2, no. 1, p. 47–65.

[133] Zhou, W.-H., Lee, J., Li, G.-L. and Imai, K. Embedding Game of Life into a Simple Asynchronous Cellular Automaton. In: *2014 Second International Symposium on Computing and Networking*. 2014, p. 503–506.

Appendices – Paper Reprints

This part contains the main outcomes of the thesis – selected papers published by the author presenting detailed studies of the issues mentioned in Section 4.

Appendix I

Evolution of Cellular Automata Using Instruction-Based Approach

BIDLO Michal a VAŠÍČEK Zdeněk

In: *2012 IEEE World Congress on Computational Intelligence*. CA: Institute of Electrical and Electronics Engineers, 2012, pp. 1060-1067. ISBN 978-1-4673-1508-1.

M. Bidlo proposed the main idea, designed the experiments and wrote cca 75% of the paper, Z. Vašíček contributed technically to the design and execution of the experiments, processing the results and writing cca 25% of the paper.

Evolution of Cellular Automata Using Instruction-Based Approach

Michal Bidlo
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, 61266 Brno
Czech republic
Email: bidlom@fit.vutbr.cz

Zdenek Vasicek
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, 61266 Brno
Czech republic
Email: vasicek@fit.vutbr.cz

***Abstract*—This paper introduces a method of encoding cellular automata local transition function using an instruction-based approach and their design by means of genetic algorithms. The proposed method represents an indirect mapping between the input combinations of states in the cellular neighborhood and the next states of the cells during the development steps. In this case the local transition function is described by a program (algorithm) whose execution calculates the next cell states. The objective of the program-based representation is to reduce the length of the chromosome in case of the evolutionary design of cellular automata. It will be shown that the instruction-based development allows us to design complex cellular automata with higher success rate than the conventional table-based method especially for complex cellular automata with more than two cell states. The case studies include the replication problem and the problem of development of a given pattern from an initial seed.**

***Index Terms*—Cellular automaton, development, replication, evolutionary design.**

I. Introduction

In the recent years cellular automata (CA) have been successfully applied in many scientific areas. The development of a cellular automaton usually represents a complex process during which a non-trivial global behavior based only on local cell interactions using simple rules may emerge [1]. However, the design of a transition function according to which the CA should develop to solve a given problem is a challenging task. The problem is that the number of possible solutions grows exponentially with the increasing number of cell states and the size of the cellular neighborhood. Moreover, the process of creating the transition function is less intuitive than the traditional algorithm design because of local cell interactions and parallel matter of the CA development. Therefore, non-traditional approaches have often been applied, including evolutionary algorithms.

The goal of this paper is to introduce an instruction-based approach for the development of cellular automata. The main idea is to represent the transition function by a program (a sequence of instructions performing simple elementary operations) rather than by a table specifying a new state of a cell for all the possible combinations of states in the cellular neighborhood. It will be shown that by using the instruction-based approach the transition function for a given problem may be designed in substantially shorter time and with higher success rate in comparison with the conventional (table-based) approach. The experiments performed to demonstrate the ability of the proposed approach consider the replication problem and the development of a specified pattern in the cellular automaton. The simple genetic algorithm will be utilized to design the cellular automata.

The paper is organized as follows. The rest of this section briefly introduces the basic principles of cellular automata and summarizes the related work. In Section II the concept of instruction-based development for cellular automata is described. The setup of the evolutionary system utilized for the experiments is stated in Section III. Overview of the experimental results and discussion is proposed in Section IV. Finally, Section V provides concluding remarks and possible direction of future research.

A. Cellular Automata

Cellular automata, originally invented by Ulam and von Neumann in 1966 [2], represent a mathematical model intended to study the behavior of complex systems, especially the questions of whether computers can self-replicate. Cellular automata may also be considered as a biologically inspired technique to model and simulate the cellular development. A two-dimensional (2D) cellular automaton consists of a regular grid of cells, each of which can occur in one state from a finite set of states. The states are updated synchronously in parallel according to a local transition function. The synchronous update of all the cells of the CA is called a developmental step. The next state of a cell depends on the combination of states in the cellular neighborhood. In this paper the cellular neighborhood will be considered as a 5-tuple comprising the investigated cell and its immediate neighbor in the north, south, east and west direction. The standard form of the transition function defines next state of a given cell for every possible combination of states in its neighborhood. Let us denote $s_N s_S s_E s_W s_C \rightarrow s_{Cnew}$ a rule of the transition function, where s_N, s_S, s_E, s_W and s_C represents the actual state of the north, south, east, west and the central cell in the cellular neighborhood respectively and s_{Cnew} denotes the next

state of the investigated (central) cell. This concept is referred to as von Neumann's cellular neighborhood consisting of 5 cells. Boundary conditions have been considered for a finite size of the cellular grid. Typically zero boundary conditions have been applied which means that the non-existing neighbors of the cells at the grid boundary are considered as cells in state 0. Another case may involve cyclic boundary conditions, i.e. the opposite cells at the grid boundary are considered to be neighbors and then the 2D CA can be viewed as a toroid. In case of uniform cellular automata the transition function is identical for all the cells. In general, non-uniform CA may have each cell driven by different transition function.

In this paper 2D uniform cellular automata with von Neumanns neighborhood and cyclic boundary conditions will be considered.

B. Related Work

Cellular automata have been applied to solve many complex problems in different areas. A detailed survey of the principles and analysis of various types of cellular automata and their applications is summarized in [1]. Sipper [3] investigated the computational properties of cellular automata and proposed an original evolution-based method called cellular programming for the design of non-uniform cellular automata. He demonstrated the success of this approach in solving some typical problems related to the cellular automata, e.g. synchronization task, ordering task or the random number generation. In the recent years, scientists have been interested in the design of cellular automata for solving different tasks using the evolutionary algorithms.

Several works dealt with the replication problem in the past as well as in the recent years. Many works have dealt with the design and development of cellular automata or more general cell-based systems (e.g. Random Boolean Networks [4]). For example, Miller investigated the problem of evolving a developmental program inside a cell to create multicellular organisms of arbitrary sizes and characteristics. He presented a system in which the organism organizes itself into a well defined patterns of differentiated cell types (e.g. the French flag) [5]. Kowaliw et al. proposed a simplified model of biological embryogenesis instantiating a subset of 2D cellular automata and a methodology for "growing" the cells into agents utilizing only local interactions. His approach was called Bluenome Developmental Model [6]. Tufte and Haddow utilized a FPGA-based platform of Sblocks [7] for the online evolution of digital circuits. The system actually implements a cellular automaton whose development determines the functions and interconnection of the Sblock cells in order to realize a specified behavior [8]. The rules for the development of the cellular automaton has been designed by evolutionary algorithm. Considering the popular replication problem, probably the most known approach represents the Langton's self-replicating loops [9] that utilize special instructions encoded in the cell states to determine the development steps of the cellular automaton. In particular, the loop starts its replication by creating a "construction arm" by means of which the new copy of itself emerges. The instruction specified by the combinations of states in this arm determines the next step of the replication process (including turns, loops closing and starting the next replication process). Pan and Regia also studied the replication in cellular automata [10]. However, they adopted a uniform tree-based approach based on Genetic Programming for representing both arbitrary cellular automata structures and the rules that control the cell's transitions. As the authors state "There is no identifiable instruction sequence or construction arm, the replicating structures generally translate and rotate as they reproduce, and they divide via a fissionlike process that involves highly parallel operations." [10]. We found their approach very inspirative because it actually introduces new way of determining the states during the CA development. However, we also felt that the method utilized to calculate the transition function might be simplified substantially by introducing elementary operations and suitable encoding with respect to the form of the cellular neighborhood. As we demonstrate, our approach is applicable on different problems in two-dimensional cellular automata.

II. Instruction-Based Development for Cellular Automata

The instruction-based development (IBD) was originally introduced in [11] as an advanced generative genotype–phenotype mapping in the evolutionary design. The main goal was to provide an evolutionary system for the automatic development of generic solutions for different problems. The instruction-based approach demonstrated its ability to reduce the search space allowing to develop (arbitrarily) large structures (instances) of digital circuits.

However, the concept of instructions also may be utilized for effective representation of functions (similarly to Genetic Programming for the evolution of computer programs [12]). Cellular automata belong to the systems in which an efficient calculation of the local transition function (determining the process of their development) is essential to solve a given problem. Conventionally the local transition function is represented by a table that specifies the next state of a cell for all the possible combinations of states in its neighborhood. In case of increasing the number of cell states the number of such combinations grows exponentially and thus the representation and design of the transition function becomes difficult. It may me possible to specify implicit rules of the transition function (e.g. for some combinations of states the new state of the cell does not change) but the problem is how to determine the set of implicit rules for a given task. Therefore, we will represent the transition function by a program whose instructions perform elementary or more complex operations over the cell states of the cellular neighborhood and the next state is chosen deterministically from this modified neigborhood. Whilst in [11] the instructions were intended to manipulate the circuit building blocks (i.e. to perform a construction process), in this paper another instruction set has to be chosen. In particular, the instructions will be devoted to the calculation over cell states and other operations related to the cellular neighbor-

hood. The main idea is to demonstrate that the instruction-based approach combined with evolutionary algorithms may be widely applicable. In this paper the case studies include some problems of cellular automata development, specifically the replication problem and the development of a given pattern from an initial seed. The objective is to show that if a suitable set of instructions is utilized for the evolution of a program-based transition function of a cellular automaton, then a given behavior of the CA can be achieved with higher success rate in comparison with the conventional table-based representation.

A. Operations on the cellular neighborhood

The goal of the IBD approach to cellular automata evolution is to provide a technique for efficient updating the cell states during the CA development with respect to the states of the neighboring cells. The operations of the instructions have been chosen with respect to the form of the cellular neighborhood. The execution of the program allows to modify the states in the cellular neighborhood and subsequently to determine the next state of the investigated cell. The following development algorithm will be considered for each cell of the CA:

1) Copy the cell states of the cellular neighborhood into a temporary data structure whose form corresponds to the cellular neighborhood.
2) Execute the program representing the transition function whose instructions will modify the states in the temporary data structure.
3) Return the state of the central cell in the temporary data structure as the next state – the result of the transition function.

The set of instructions that may be utilized in the program calculating the transition function is summarized in Table I. As evident the instructions include operations that can modify one or more cells in the neighborhood copy and the empty operation allowing to alter the efficient length of the program during the evolutionary process. Since the instructions operate over the copy of the cellular neighborhood in a temporary data structure, the process of calculation of the next state of a cell does not influence the states of other cells during a development step and therefore the next states of all the cells can be determined in parallel which is a characteristic feature of cellular automata. The instructions were chosen with respect to general operations that are possible to perform over cell states (i.e. logic and arithmetic operations over the state values, transfer a cell state to a different cell in the neighborhood, swapping the states of two neighbors etc.). However, no advanced optimization of the instruction set has been performed in this stage of research because the selection of proper instructions for a given CA behavior represents a difficult task and in many cases is a subject of experimental work.

TABLE I
THE SET OF INSTRUCTIONS UTILIZED FOR THE DEVELOPMENT OF CELLULAR AUTOMATA. $N[i_1], N[i_2]$ DENOTE THE CELL STATES FROM THE NEIGHBORHOOD POSITIONS DETERMINED BY THE INSTRUCTION ARGUMENTS i_1, i_2, S REPRESENTS THE NUMBER OF CELL STATES AND N, S, E, W AND C SPECIFIES THE CELL STATE IN THE NORTH, SOUTH, EAST, WEST AND CENTRAL POSITION IN THE NEIGHBORHOOD RESPECTIVELY.

Instruction	Operation	Description
AND	$N[i_1] = N[i_1] \wedge N[i_2]$	logic AND
OR	$N[i_1] = N[i_1] \vee N[i_2]$	logic OR
XOR	$N[i_1] = N[i_1] \oplus N[i_2]$	logic XOR
NOT	$N[i_1] = notN[i_1]$	bitwise NOT
INV	$N[i_1] = S - N[i_1]$	inverse state
MIN	$N[i_1] = min(N[i_1], N[i_2])$	minimum
MAX	$N[i_1] = max(N[i_1], N[i_2])$	maximum
SET	$N[i_1] = N[i_2]$	replace
INC	$N[i_1] = N[i_1] + 1$	increment
DEC	$N[i_1] = N[i_1] - 1$	decrement
SWP	$N[i_1] \leftrightarrow N[i_2]$	swap
ROR	$WCE \rightarrow EWC$	rotate right
ROL	$WCE \rightarrow CEW$	rotate left
ROU	$UCS \rightarrow CSU$	rotate up
ROD	$UCS \rightarrow SUC$	rotate down
NOP		no operation

B. Properties of the Instruction-Based Transition Function

If an evolutionary algorithm is applied to design a CA, the instruction-based approach is able to shorten the chromosome substantially and therefore to reduce the search space. In fact, the design of a CA consists of the evolution of its local transition function.

For example, if a transition function ought to be evolved for a CA working with 4 cell states (that is used in some of the experiments presented in Section IV), then the fully defined table-based transition function consists of $4^5 = 1024$ integers (it is the length of a chromosome representing the complete table of the transition function). Therefore, there are in total $4^{1024} = 3.2317 \times 10^{616}$ different transition functions for this CA which represents the search space of the evolutionary algorithm. Consider that the IBD approach is utilized and the goal is to evolve a program consisting of 10 instructions. Moreover, assume that a single instruction consists of 3 integers (operation code and two arguments), there are 16 different instructions (i.e. 16 different operation codes) and each of the arguments can posses one of 5 different values. Then the length of a chromosome is $10 \times 3 = 30$ integers and the size of the search space consists of $(16 \times 5 \times 5)^{10} = 1.048576 \times 10^{26}$ different programs which is substantially less in comparison with the table-based representation.

As stated in the previous section, the program is executed over a copy of the cellular neighborhood. Therefore, the next states of all the cells can be calculated independently (in parallel) as usual in common (synchronous) cellular automata. Another important aspect of the IBD approach is that the process of calculating the next state for a given cell is deterministic (there is a specific combination of states in the neighborhood copy which the program operates on, each instruction of the program performs a deterministic operation (function) modifying the states in the neighborhood and the resulting value — next state — is always considered in a specific cell of the neighborhood after executing the program). Considering this feature, the instruction-based transition function can be deterministically transformed to a corresponding

table-based transition function without changing the nature of cellular automata.

III. Evolutionary System Setup

The simple genetic algorithm (GA) was utilized for the evolutionary design of the cellular automaton that exhibits the specified behavior. For the comparison purposes we consider the evolution of common table-based local transition function as well as the program-based transition function as described in the previous section. The table-based approach considers the evolution of a complete transition function (i.e. to determine a next state for all the possible combinations of states in the cellular neighborhood). In case of the IBD approach a program to be evolved consists of 10 instructions. This value was determined experimentally in order to provide a sufficient resources to calculate the next states. Of course, some of the resulting solutions use NOP instructions so the effective length of the program can be reduced. However, if shorter programs ought be evolved, then the number of correct solutions in the search space may be reduced and the success rate decreases.

In all the experiments, the population consists of 16 chromosomes which are initialized randomly (with respect to the correct range of each gene) at the beginning of evolution. The chromosomes are selected by means of the tournament operator with the base 4. The experiments showed that the crossover operator is not suitable for this problem, thus only the mutation operator is applied as follows. Two integers of the chromosome are chosen randomly and their values are mutated by generating new random values in the appropriate range.

Each candidate CA is evaluated during 30 development steps according to the transition function encoded in the chromosome. The following subsections describe the specific features of the evolutionary system with respect to the two different approaches.

The initial state of the CA, the way of calculating the fitness function and the number of generations of the evolution depends on the problem to be solved and therefore their description will be covered in Section IV.

The way of encoding the transition function in the genome for the table-based and program-based representation and its properties is described in the following subsections.

A. Table-Based Transition Function

In case of the table-based transition function the chromosome encodes the next states of a cell for all the possible combinations of states in the cellular neighborhood. The index of a given next state in the chromosome is specified implicitly by means of the value expressed by the number representing the combination of states in the cellular neighborhood. The base of this number equals the number of possible cell states. Therefore, if we consider the general form of the rule $s_N\ s_S\ s_E\ s_W\ s_C \rightarrow s_{Cnew}$, only the part on the right of the arrow are encoded in the chromosome. For example, if a cellular automaton ought to be evolved working with 2 different cell states and von Neumann's neighborhood consisting of 5 cells, there are 2^5 rules of the local transition function. Consider the rule $1\ 0\ 0\ 0\ 1 \rightarrow 0$. Since the combination of states $1\ 0\ 0\ 0\ 1$ corresponds to the binary representation of value 17, the output value (0) will be placed in the chromosome at the position 17.

B. Program-Based Transition Function

The program-based representation of the transition function is encoded in the chromosome as a finite sequence of instructions from Table I. Each instruction is encoded as three integers (operation code and two arguments) whose value ranges depend on the number of instructions and the meaning of their arguments. The main advantage of this approach is that the length of the genome is independent on the number of cell states and the size of the cellular neighborhood. Therefore the search space can be reduced substantially.

IV. Experimental Results and Discussion

The abilities of the proposed instruction-based development approach introduced in the previous sections will be demonstrated on two problems: (1) the replication problem and (2) the problem of development of a given pattern from a seed. The experimentsl results and discussion are given in this section.

A. Replication Problem

The goal of the replication problem is to obtain a copy of a given structure in a finite number of development steps. The structure is represented by the initial state of the CA. The genetic algorithm is applied to design a transition function by means of which the CA develops so that there is a given number of copies of the initial structure after a finite number of development steps. The set of experiments performed in this section considers searching for the transition function (in the form of table and program) for the replication of structures of different complexity and size. As noted in Section I-B there are several approaches to the replication problem. Probably the simplest technique able to replicate an arbitrary structure is based on additive cellular automata rules [1]. This problem can be viewed as a basis for investigating the abilities of the proposed method (having a known solution, we may search for the same or similar transition functions using the conventional and proposed approach).

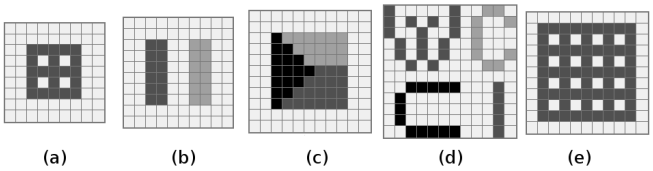


Fig. 1. Patterns considered in the experiments.

Five sets of experiments were performed, each of which contained 100 independent runs of the GA. The first set considered the replication of a simple grid structure (Figure 1a), the second was devoted to the replication of French flag pattern (Figure 1b), the third set is the replication of Czech flag

Appendix I

```
best_fitness = 0  # fitness out of all development steps
const REPLICS = the num. of required copies of the given pattern

initialize the CA by the pattern to be replicated
FOR int step = 1 TO DEVEL_STEPS DO
{
   fitness = 0  # fitness in one development step
   replicas_cnt = 0  # num. of replicas found in a devel. step
   ca_step(cal, genome->prog);

   FOR row = 0 TO CA_HEIGHT - PATTERN_HEIGHT DO
   {
      FOR col = 0 TO CA_WIDTH - PATTERN_WIDTH DO
      {
         partial_fitness = 0  # fitness in specific part of CA
         FOR pr = 0 TO PATTERN_HEIGHT - 1 DO
            FOR pc = 0 TO PATTERN_WIDTH - 1 DO
               IF ca[row+pr][col+pc] == pattern[pr][pc] THEN
                  partial_fitness = partial_fitness + 1
         save the partial_fitness value
         IF found perfect pattern at position (row, col) THEN
            replicas_cnt = replicas_cnt + 1
      }
   }
   fit = sum of the REPLICS best saved partial fits
   # add a bonus if the solution produces more replicas
   fit = fit + replicas_cnt * PATTERN_HEIGHT * PATTERN_WIDTH

   # save the best fitness out of all development steps
   IF fitness > best_fitness THEN
      best_fitness = fitness
}

RETURN best_fitness
```

Fig. 2. Calculating the fitness function for the replication problem. The pattern dimensions $PATTERN\_WIDTH$ and $PATTERN\_HEIGHT$ also include a border consisting of a single line of inactive cells (cells in state 0) because we require the replicated structures to be separated each other.

(Figure 1c), the fourth and fifth replicate WCCI abbreviation (Figure 1d), where at least 3 and 4 copies are required respectively. The evolution was executed independently for the design of the conventional table-based transition function and the program-based representation. The algorithm calculating the fitness function is shown in Figure 2 and its principle can be described as follows. After each development step, every part of the CA is explored by comparing the states of the given pattern with the appropriate cell states at the corresponding positions in the CA. If a state match is detected, then a partial fitness value associated with the specific part of the CA is increased by one. After exploring the part of the CA the resulting partial fitness is saved into a temporary array. If the partial fitness equals the number of cells the replicating pattern is composed of, then the value of a replicas counter variable is increased by one. After exploring all the parts of the CA, the replicas counter contains the number of perfectly matched patterns (i.e. the number of replicas that emerged after a given development step). The fitness value of the given development step is calculated as the sum of the $REPLICS$ best partial fitness values, where $REPLICS$ represents the number of required copies of the (initial) pattern to be replicated. If the replicas counter detected at least one replicated pattern, then the fitness value is increased appropriately to prefer solutions that are able to create perfect replicas. As a final fitness value of the CA (i.e. the fitness of the candidate transition function) is considered the highest fitness from all the development steps during which the CA was evaluated.

The results of the replication experiments are summarized in Table II. For each pattern the success rate and the average number of generations needed to find a perfect solution were measured. The proposed program-based transition function overcomes the conventional approach in all presented cases,

TABLE II
STATISTICAL RESULTS FOR THE REPLICATION PROBLEM CONSIDERING THE INSTRUCTION-BASED AND TABLE-BASED DEVELOPMENT. THE GRID STRUCTURES (FIG. 1A) WERE DEVELOPED IN CA WITH 2 CELL STATES, THE OTHER PATTERNS CONSIDERED 4 CELL STATES. IF NOT EXPLICITLY SPECIFIED, 3 REPLICAS WERE REQUIRED.

Instruction-based development						
Pattern	**Succ** [%]	**Number of generations** avg.	std. dev.	min.	median	max.
Fig. 1a	100.0	23	19.4	1	18	80
Fig. 1b	100.0	22	16.4	1	19	80
Fig. 1c	100.0	22	21.4	1	18	112
Fig. 1d	100.0	23	18.1	1	18	81
Fig. 1d (4 repl.)	100.0	55	43.6	2	47	256
Table-based development						
Fig. 1a	9.0	5634	2490.2	1500	5250	9052

especially for more complex patterns. In case of the grid structure replication a perfect program was evolved in all runs, whilst the table-based approach succeeded only in 9% of evolutionary runs. It is important to note that the table-based approach did not provide any solution to the remaining patterns considered in the experiments. We assume that this result is caused by the cardinality of the search space that is substantially higher for the table-based representation and the evolution is not able to explore it effectively. Another aspect of this issue is probably based on the operations needed to express the local transition function. In case of the table-based representation, the transition function actually needs to be created at a low level (i.e. for every combination of states in the cellular neighborhood a new state has to be specified). However, if the instruction-based approach is considered, the new state is calculated using higher-level operations (like in a common programming language), the corresponding program can be shorter in comparison with the complete table which leads to a reduction of the search space and the evolution is able to explore it more effectively. We determined that several different programs were evolved that produces at least 3 perfect replicas of the given pattern. Although we required 3 replicas, the evolution found in some case a solution providing 4 replicas of the structure.

An example of evolved solution is shown in Figure 3. In addition, the table-based transition function produced two solutions for 3 replicas that exhibit a couple of extra active cells between the replicated patterns (Figure 4). This behavior was not observed in the program-based approach (only pure replicated structures were generated). It is probably caused by the fact that the evolution of the table-based representation directly allows to alter each single output state of the transition function whilst the program-based approach actually represents an indirect mapping between the input combinations of states in the cellular neighborhood and the output states. This feature may be considered as both advantage and disadvantage of the program-based approach. The benefit lies in the fact that the program-based solutions produce perfect outputs without

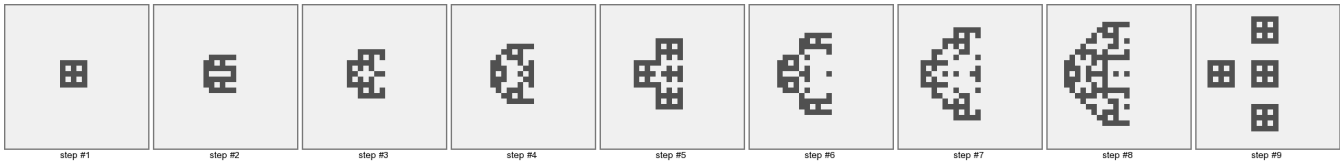


Fig. 3. Development of evolved cellular automaton for the replication of a grid structure.

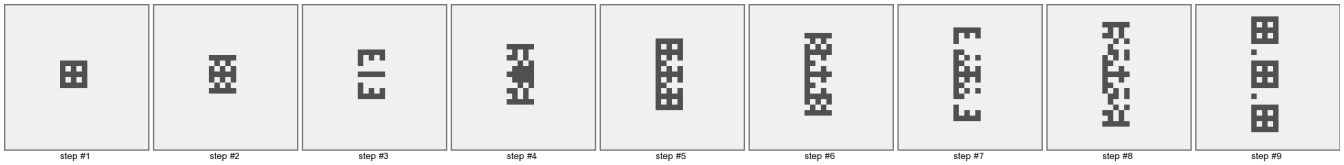


Fig. 4. Example of replication of a simple grid structures with additional active cells.

undesirable active cells. On the other hand the drawback is that more complex transition functions require (as expected) more instructions in the program. Nevertheless the evolution is able to tackle that very efficiently because the proposed approach solved all the considered problems with substantially higher success rate and lower computational effort in comparison with the table-based transition function.

Another examples illustrating the replication of more complex irregular patterns (the Czech flag and a WCCI pattern) are illustrated in Figure 5 and 6. Both of these automata operate with 4 cell states.

The evolved transition functions exhibit the features of additive rules described in [1]. Several different variants were obtained differing in the number and direction of replicas with respect to the position of the initial structure. In addition to the pattern used during the evolution, the resulting programs are in many cases able to replicate different structures which confirms the properties of the replicators mentioned in [1].

B. Pattern Development Problem

Another issue that was investigated in our experiments is the problem of the development of a given pattern in a cellular automaton from a seed. It means that the initial state of the CA is represented only by the central cell in non-zero state, all the other cells possess the state 0. During the evolutionary process the CA is examined if it matches with the specified pattern after each development step. In these experiments the dimensions of the cellular automaton correspond to the dimensions of the pattern that ought to be developed. The goal is to design a transition function (again, in the form of table and program) according to which the CA develops from the seed into the given pattern.

Four sets of experiments were performed. The first pair of experiments considered the development of a grid structure consisting of 5x5 cells (Figure 1a)) and 9x9 cells (Figure 1e). In the second pair of experiments French flag ought to be developed (Figure 1b) with the dimensions 6x6 and 9x9 cells. The candidate solutions are evaluated as follows. A partial fitness value is calculated after each development step as the number of cells of the CA whose state equals the state of the corresponding cell of the target pattern. The fitness function of a candidate transition function is evaluated as the maximum of the partial fitness values from all the development steps.

TABLE III
STATISTICAL RESULTS FOR THE PATTERN DEVELOPMENT PROBLEM CONSIDERING THE INSTRUCTION-BASED AND TABLE-BASED APPROACH. THE GRID STRUCTURES (FIG. 1E) WERE DEVELOPED IN CA WITH 2 CELL STATES, THE OTHER PATTERNS CONSIDERED 4 CELL STATES.

Instruction-based development						
Pattern	**Succ. [%]**	**Number of generations**				
		avg.	**std. dev.**	**min.**	**median**	**max.**
Fig. 1a	100.0	14358	16711.5	143	7543	86445
Fig. 1e	60.0	32504	23387.7	2888	24828	89228
Fig. 1b	79.0	37925	27117.1	1211	31991	97717
French9x9	23.0	62095	21979.8	18784	62143	90233
Table-based development						
Fig. 1a	100.0	402	757.3	19	118	4075
Fig. 1e	76.0	24331	26200.6	118	16353	96980
Fig. 1b	54.0	28896	26264.1	475	22028	92948
French9x9	1.0	30614	0.0	30614	30614	30614

Table III summarizes the statistical results from the experiments mentioned in the previous paragraph. The evolution succeeded in all cases and provided solutions that perfectly fulfil the objectives specified in the fitness function. There are some interesting facts that were observed in both representations of the transition function. The first is that the instruction-based approach exhibits higher success rate in most sets of experiments. The only case in which the table-based representation is more successful is the development of a 9x9 grid structure (the program-based approach succeeded in 60% whilst the conventional method in 76%). This issue can be explained as follows. The problem considers a binary CA whose 5-neighborhood implies 2^{32} possible transition functions specified by the table (the chromosome consists of 32 bits). However, the search space of the program-based approach is in this case substantially bigger. For example, if 10 instructions in the programs are considered, each consisting of 3 integers, there are 30 integers in the chromosome, each of which can possess at least 5 different values so the search

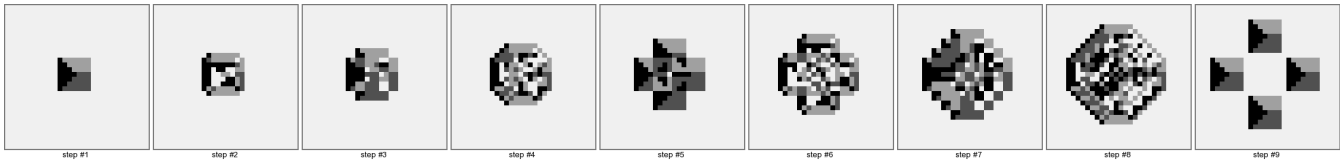


Fig. 5. Development of evolved cellular automaton for the replication of the Czech flag.

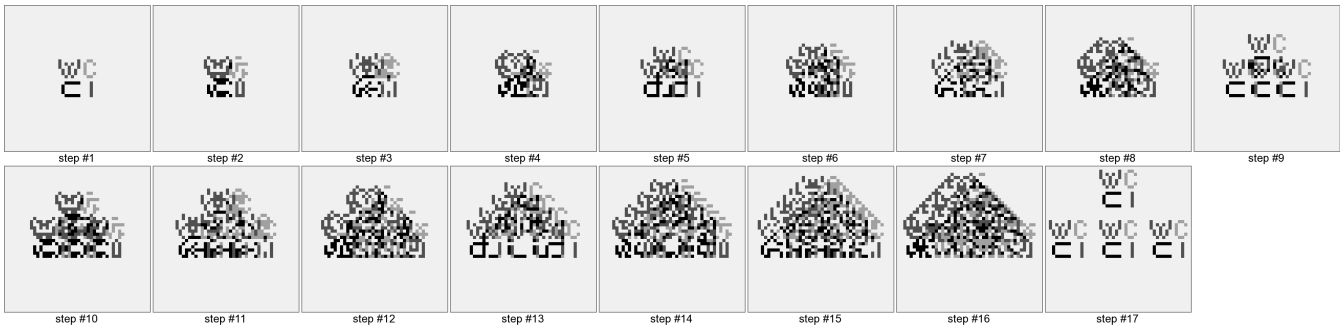


Fig. 6. Development of evolved cellular automaton for the replication of WCCI structure.

space contains at least 5^{30} candidate solutions. Therefore it is harder to find a working solution for the 9x9-cell structure in so big search space. The second interesting issue is that although the program-base approach mostly exhibits higher success rate, the computational effort (expressed by the number of generations needed to evolve a working solution) is higher than in case of the conventional approach. This fact was observed in all the experiments performed in the pattern development problem. We assume that this feature is caused by more complex (indirect) mapping between a program and the corresponding output states of the transition function of the cellular automata.

Figure 7 shows an example of evolved solution for the development of French flag in a cellular automaton. In this case we obtained several solutions that differ in the behavior of the developed structure if the CA continues to develop. In most cases the French flag pattern represents an intermediate state of the CA that is totally destroyed if the development continues. The second group of solution is able to periodically recreate the given pattern and the last case includes several solutions that produce the French flag that is stable during the subsequent development of the CA. These classes of solutions are expectable. Since the CA possesses finite dimensions and the number of cell states is also finite, it can not exhibit infinite development through infinite different states. Therefore, if the CA does not exhibit a stable pattern after a finite number of development steps, then it generates a finite number of different patterns in a loop (e.g. see Figure 7)). The corresponding program that was found by evolution is shown in Table IV. It is very difficult to identify the principle of this program (similarly as to identify the individual rules of a transition function) because the CA behavior is an emergent property of interaction of all the cells. It can be observed that all the (temporary) neighborhood cells are affected by the program so that the development of French flag is probably not a trivial task. Note that the exact French flag pattern was reached only in CA whose dimensions correspond to the pattern size. In larger CA, although, it is possible to develop the pattern in a subpart of the CA but some of the other cells are affected too that surrounds the target pattern immediately (confirmed by the experiments).

TABLE IV
EVOLVED 6X6 CELLULAR AUTOMATON PROGRAM FOR THE DEVELOPMENT OF FRENCH FLAG PATTERN. THE EVOLUTION WORKED WITH 10-INSTRUCTION PROGRAM, THE RESULTING SOLUTION CONTAINED 2 NOPS THAT WERE SUBSEQUENTLY REMOVED.

Line num.	Instruction
1:	MAX W C
2:	XOR C N
3:	MIN S E
4:	ROD
5:	AND E S
6:	DEC E
7:	OR C E
8:	XOR C W

V. CONCLUSIONS

In this paper we presented an instruction-based approach to the development of 2D cellular automata and their design using genetic algorithm. The idea was to shorten the genotype and reduce the search space especially for the CAs with more than 2 cell states. Two problems were considered in order to demonstrate the abilities of the proposed approach: (1) the replication problem and (2) the problem of development of a given pattern from a seed.

In case of the replication problem, the instruction-based approach overcame the conventional table-based transition function in all the performed experiments. We determined that in addition to the perfect success rate this method also reduces

Appendix I

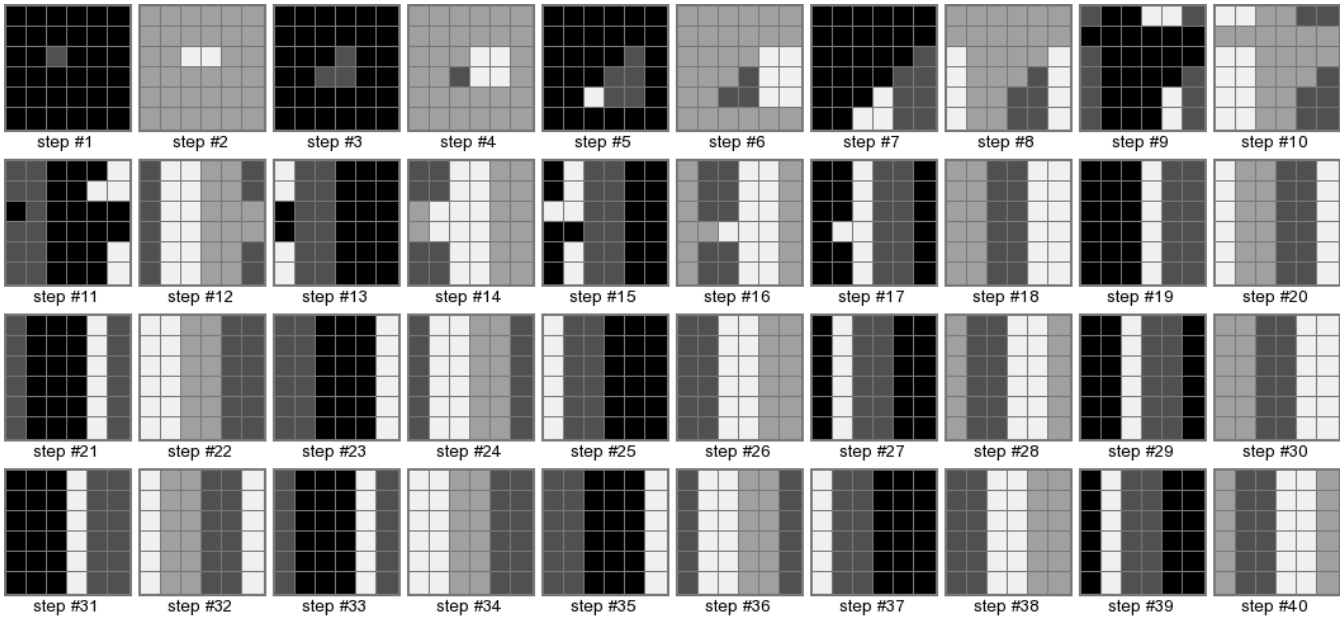


Fig. 7. Development of French flag pattern in a cellular automaton. This solution shows the development process in which the French flag emerges for the first time in step 26. Then the pattern is destroyed and emerges again with the period of 12 development steps (the next instance can be observed in step 38).

the computational effort needed to evolve a working solution of the replication problem.

The pattern development from a seed proposed interesting results in both of the instruction-based method and the conventional approach. Whilst the instruction-based development exhibits substantially higher success rate in most of the experiments, the conventional approach provides lover computational effort for obtaining a working solution.

In summary the proposed method works very well for more complex cellular automata, even for those in which no working solution was found by means of the conventional approach. We assume that the instruction-based approach is applicable to many other problems whose solution can be realized using cellular automata. The experiments that were performed in this paper represent problems for which successful solutions are known. However, we are going to experiment with more applications in order to determine the cellular automata behavior in different conditions. For example, the optimization of instruction set for a specific CA behavior seems to be an interesting area. Experiments in other application domains are in progress (e.g. development of computational structures or image operators may represent suitable candidates).

ACKNOWLEDGMENT

This work was supported by the Czech science foundation projects P103/10/1517 and GD102/09/H042, the research programme MSM 0021630528, the BUT projects FIT-S-11-1, FIT-S-12-1 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] S. Wolfram, *A New Kind of Science*. Champaign IL: Wolfram Media, 2002.

[2] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[3] M. Sipper, *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194*. Berlin: Springer-Verlag, 1997.

[4] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[5] J. F. Miller, "Evolving developmental programs for adaptation, morphogenesis and self-repair," in *Advances in Artificial Life. 7th European Conference on Artificial Life, Lecture Notes in Artificial Intelligence, volume 2801*. Dortmund DE: Springer, 2003, pp. 256–265.

[6] T. Kowaliw, P. Grogono, and N. Kharma, "Bluenome: A novel developmental model of artificial morphogenesis," in *Proc. of the Genetic and Evolutionary Computation Conference, GECCO 2004, Lecture Notes in Computer Science, part I., volume 3102*. Springer-Verlag, 2004, pp. 93–104.

[7] P. C. Haddow and G. Tufte, "Bridging the genotype–phenotype mapping for digital FPGAs," in *Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware*. Los Alamitos, CA, US: IEEE Computer Society, 2001, pp. 109–115.

[8] G. Tufte and P. C. Haddow, "Towards development on a silicon-based cellular computing machine," *Natural Computing*, vol. 4, no. 4, pp. 387–416, 2005.

[9] C. G. Langton, "Self-reproduction in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1–2, pp. 135–144, 1984.

[10] Z. Pan and J. A. Reggia, "Computational discovery of instructionless self-replicating structures in cellular automata," *Artificial Life*, vol. 16, no. 1, pp. 39–63, 2010.

[11] M. Bidlo and J. Škarvada, "Instruction-based development: From evolution to generic structures of digital circuits," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 12, no. 3, pp. 221–236, 2008.

[12] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

Appendix II

Evolution of Cellular Automata with Conditionally Matching Rules

BIDLO Michal a VAŠÍČEK Zdeněk

In: *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*. Cancún: IEEE Computer Society, 2013, pp. 1178-1185. ISBN 978-1-4799-0452-5.

Conference CORE rank in the year of publication: A; M. Bidlo proposed the main idea, designed the experiments and wrote cca 75% of the paper, Z. Vašíček contributed technically to the design and execution of the experiments, processing the results and writing cca 25% of the paper.

2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México
Appendix II

Evolution of Cellular Automata with Conditionally Matching Rules

Michal Bidlo
Brno university of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2
61266 Brno, Czech Republic
Email: bidlom@fit.vutbr.cz

Zdenek Vasicek
Brno university of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2
61266 Brno, Czech Republic
Email: vasicek@fit.vutbr.cz

***Abstract*—This paper introduces a method of representing transition functions for the purposes of evolutionary design of cellular automata. The proposed approach is based on conditions specified in the transition rules that have to be satisfied in order to determine the next state of a cell according to a specific rule. The goal of this approach is to reduce the number of elements needed to represent a transition function while preserving the possibility to specify traditional transition rules known from the conventional table-based representation. In order to demonstrate abilities of the proposed approach, the replication problem and pattern transformation problem in cellular automata will be investigated. It will be shown that the evolution is able to design transition functions for non-trivial behavior of two-dimensional cellular automata that perfectly fulfil the specified requirements.**

I. Introduction

Cellular automata (CA) represent a biologically inspired computational model in which time and space are discrete. The cells represent basic computational elements whose states are considered as a means for storing and processing information inside the CA. Their concept was originally invented by Ulam and von Neumann in 1966 [1] in order to study the behavior of complex systems, especially the questions of whether computers can self-replicate. A two-dimensional (2D) cellular automaton consists of a regular grid of cells, each of which can occur in one state from a finite set of states. In each developmental step of the CA, the states are updated synchronously in parallel according to a local transition function. The next state of a given cell depends on the state of this cell and the combination of states in its neighborhood.

For the purposes of this paper the following basic concept of cellular automata will be considered. The cellular neighborhood is represented by a 9-tuple and consists of the investigated (central) cell and its immediate neighbors in the horizontal, vertical and both diagonal directions. This concept is referred to as Moore neighborhood and is illustrated in Figure 1. The common form of the transition function defines the next state of a given cell for every possible combination of states in its neighborhood. Let us denote $NW\ N\ NE\ W\ C\ E\ SW\ S\ SE \rightarrow C_{new}$ a rule of the transition function, where the symbols on the left of the arrow (corresponding to the cells from Figure 1) represent actual cell states of the Moore neighborhood and C_{new} denotes the next state of the investigated cell. Boundary conditions will be considered because the implementation of cellular automata considers a finite size of the cellular array. For that reason, zero boundary conditions will be applied which means that the non-existing neighbors of the cells at the boundary of cellular structure are considered as cells in state 0. Every cell will determine its next state according to a single transition function, i.e. it is a case of uniform cellular automaton.

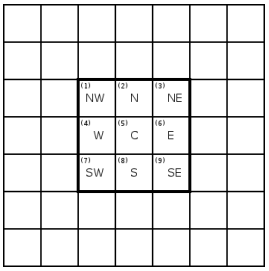


Fig. 1. Structure of Moore neighborhood in a cellular automaton. The neighborhood includes the investigated (central) cell C and its immediate neighbors. The numbers in parentheses denote the indices of cells in the neighborhood that will be considered in this paper.

Wolfram studied the mathematical fundamentals of cellular automata and analyzed their behavior from the theoretical point of view. Moreover, he performed a broad survey of various cellular automata applications and summarized the results in [2]. Sipper studied, among others, non-uniform cellular automata and proposed a specific evolutionary algorithm called Cellular Programming for the automatic design of non-uniform CA. Cellular Programming involves a population of local transition functions whose evolution (using the genetic operations of crossover and mutation) is carried out with respect to the arrangement of cells in the CA and their local interactions. Sipper demonstrated the success of this approach in solving some typical problems related to cellular automata, e.g. synchronization task, ordering task or random number generation. A hardware accelerator of Cellular Programming was also proposed [3]. Several works have dealt with evolving cellular automata using genetic algorithms and similar evolutionary techniques. Miller investigated the problem of evolving a developmental program inside a cell to create multicellular organisms of various sizes and characteristics. He presented

978-1-4799-0454-9/13/$31.00 ©2013 IEEE

a system in which the organism organizes itself into a well defined patterns of differentiated cell types (e.g. the French flag) [4]. Tufte and Haddow utilized an FPGA-based platform for online evolution of digital circuits. Their approach is based on a special architecture called Sblock that implements the cell functionality [5]. The interconnection of Sblocks in the FPGA actually implements a cellular automaton whose development determines the functions and interconnection of the individual Sblock cells in order to realize a specified behavior [6]. The rules for the development of the Sblocks has been designed using evolutionary algorithm. Kowaliw et al. proposed a simplified model of biological embryogenesis instantiating a subset of 2D cellular automata and a methodology for "growing" the cells into agents utilizing only local interactions. The Bluenome Developmental Model, as the authors denote this approach, implements a grid of cells, each of which contains a single piece of DNA-like data, which it interprets to decide its next action. The rules of this model have been searched using genetic algorithm [7]. Sometimes the traditional concept of cellular automata has been adapted to solve a specific task. For example, Random Boolean Networks represent a more general approach to the development of cellular structure in which the neighborhood of each cell is not limited to immediate neighbors only but can be specified arbitrarily. Each cell can even have different form of neighborhood. This concept was originally developed as an abstract model for studying the dynamics of gene regulation [8]. Similar technique related to the genetic regulatory networks was proposed by Dellaert et al. [9][10]. The authors implemented the process of gene regulation using Boolean functions called operons inside the genome the goal of which was to design an ambitious and extensive model of development meant that is both biologically defensible and computationally tractable.

In [11] a genetic algorithm-based approach was presented for the deign of 2D cellular structures (called agents) that act as building blocks for assembling more complex objects. The cooperation between the agents during development exhibit a self-assembling process of a target entity. The goal of this process is to produce large stable structures by evolving rules and parameters of the building blocks. Kayama has investigated a network representation of binary cellular automata rules [12]. The goal was to focus on the effective relationships between cells rather than the states themselves. This approach allows the techniques of the network theory to be used for the investigation of CA behavior. An instruction-based representation of the cellular automata rules was proposed in [13]. In this case the transition function is encoded as a program consisting of simple instructions whose aim is to modify the cellular neighborhood in order to calculate the next cell state. It was demonstrated that this approach is able to improve the process of designing cellular automata by means of genetic algorithm in comparison with the traditional encoding of the CA rules by means of a table.

The process of designing a transition function according to which the CA develops in order to achieve a specified behavior is a challenging task. The problem is that the creation of transition function is less intuitive than the traditional algorithm design because the behavior of each cell depends on its neighbors only and the cells operate in parallel during the CA development. Moreover, the number of possible transition functions grows exponentially with the increasing number of cell states and the size of the cellular neighborhood. Therefore, non-traditional approaches have often been applied both to the representation of cellular automata rules and the method of searching for a specific transition function.

This paper presents a continuation of the research introduced in [13], where an instruction-based representation of the transition function was proposed. We have determined that if a suitable form of the transition rules is applied (instead of the traditional table-based representation), it is possible to reduce the time needed to design a specific CA to solve a given task. For example, the replication problem and the problem of developing a target pattern from a seed was successfully solved by the instruction-based approach when the program representing the transition function was designed by mean of genetic algorithm. The input of the program is the combination of states in the cellular neighborhood, the output is a single value representing the next state of the investigated cell [13]. However, the subsequent experiments showed that if the program-based approach is applied, then the next state actually represents a global result of the program execution and it is difficult to detect states of the cellular neighborhood for which specific or separate rules could be more suitable. Therefore, the goal of this paper is to propose a method of representing the CA rules that is (1) more efficient that the traditional table-based approach and (2) yet allows us to describe specific transition rules in a way naturally convenient to cellular automata. This method will be designated as Conditionally Matching Rules. We will show that this approach is able to solve the replication problem (as one of the typical task in CA) and the problem of a non-trivial transformation of a given initial pattern to another target pattern whose solution was not successful using the previously mentioned approaches.

The paper is organized as follows. Section II describes the idea of conditionally matching rules. The evolutionary system setup is summarized in Section III and the experimental results are given in Section IV. Concluding remarks are stated in Section V.

II. Conditionally Matching Rules for Cellular Automata

Conventionally the local transition function is represented by a table that specifies the next state of a cell for all the possible combinations of states in its neighborhood. However, if the number of cell states or the size of cellular neighborhood increases, then the number of such combinations grows exponentially and thus the representation and design of the transition function becomes very difficult. It might be possible to specify a subset of rules for the transition function (e.g. only for the combinations of states that change the state of the investigated cell) but the problem is how to determine the set of rules for a given task especially for complex cellular automata.

In order to overcome these issues, a new encoding of cellular automata rules will be introduced. Let us call this approach as Conditionally Matching Rules (CMR). The encoding of the local transition function using CMR is fundamentally inspired by the table-based representation. It means that the CMR encoding allows to specify the transition rules as usual in the table-based approach but, in addition to that, more

general rules can be formulated whose interpretation covers several common rules in a single CMR. In particular, each rule of the CMR representation consists of a conditional part and a next state. The conditional part encodes a state and a condition for every cell in the cellular neighborhood. The next state is assigned to the investigated cell if the given rule "matches" to the combination of states in its neighborhood, i.e. if all the conditions in the conditional part are satisfied. For the purposes of this paper, the following conditions will be considered in the CMR: equal (==), not equal (! =), greater or equal than (>=), less or equal than (<=) and don't care mask ($\star$). The structure of a conditionally matching rule for Moore neighborhood and its relation to this form of neighborhood is illustrated in Figure 2.

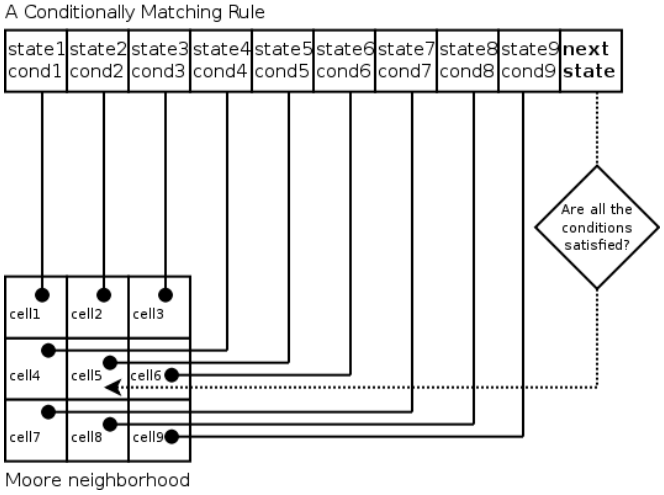


Fig. 2. Structure and interpretation of a conditionally matching rule

The local transition function of a CA consists of a finite sequence of conditionally matching rules. The process of determining the next state of a cell using the CMR-based transition function is the following. The rules are evaluated sequentially one after another. In each rule the items of the conditional part are evaluated with respect to the corresponding cell states in the cellular neighborhood. If all the conditions are satisfied, then the rule is said to match with the state of cellular neighborhood and the next state from this rule represents the result of the transition function (i.e. the new state of the investigated cell) and no more rules in the sequence need to be evaluated. If none of the rules representing the transition function matches, then the cell keeps its current state.

For example, consider a CMR-based transition function that ought to be applied to determine the next state of central cell of a given cellular neighborhood (Figure 3). In this case the transition function consists of three conditionally matching rules denoted as #1, #2 and #3. In order to determine the next state, the evaluation of the rules starts with the CMR #1. The state of cell (1) in the cellular neighborhood shown in Figure 3 satisfies the condition == 1 in the condition (1) of rule #1. Condition (2) of rule #1 is a don't care mask ($\star$) which means that this condition is also satisfied with respect to the state of cell (2). Condition (3) is also satisfied because the state of cell (3) is less than or equal to state 0 specified in this condition. However, condition (4) assumes that the

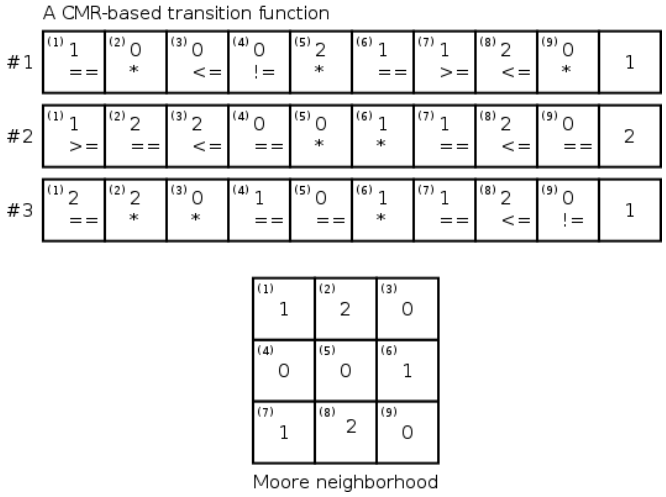


Fig. 3. Example of CMR-based transition function consisting of three conditionally matching rules

state of cell (4) does not equal 0 which is not true because the state of this cell possesses 0. Therefore, this condition is not satisfied which means that rule #1 can not match to the cellular neighborhood and can not be applied to determine the next state. The execution of the transition function continues by evaluating rule #2. As all the conditions of this rule are satisfied with respect to the corresponding cell states in the neighborhood, the next state specified in rule #2 represents the result of the transition function and hence the cell (5) will possess state 2 in the next step. In this case rule #3 does not need to be evaluated because the next state has already been determined.

Considering the concept of the CMR-based transition function, several advantageous features may be identified. Firstly, the size of representation of CMR-based transition function can be reduced in comparison with the conventional table-based format. It is based on the possibility to use relational operators (especially ! =, <=, >=) and the don't care mask ($\star$). In fact, a single CMR with some of those conditions represents several rules of the conventional table-based transition function. Secondly, the CMR approach is deterministic which is given by the convention that if a rule from the sequentially evaluated sequence matches, then its next state represents the result of the transition function, otherwise the investigated cell keeps its current state. And finally, CMR-based transition functions can be deterministically transformed to the complete table-based representation. If all the possible combinations of states are generated for a given type of cellular neighborhood, then for each combination a next state is calculated using the CMR that corresponds to a specific item in the table-based transition function. Therefore, the CMR encoding fully preserves the features of traditional cellular automata.

III. Experimental Setup

Simple genetic algorithm (GA) was utilized for the evolution of CMR-based transition function in order to achieve a specific behavior.

Several sets of experiments were performed considering various numbers of rules encoded in a chromosome. Each

chromosome represents a candidate transition function represented as a finite sequence of conditionally matching rules. The structure of each CMR is identical to that shown in the top part of Figure 2. Each CMR is encoded as a finite sequence of integers representing the conditional parts (i.e. codes of states and condition operators) and the next state.

In all experiments, the population consists of 8 individuals that are initialized randomly at the beginning of evolutionary process. The chromosomes are selected by means of tournament operator with the base 4. Each pair of selected chromosomes (parents) undergo one-point crossover with the probability 50% in order to generate two offspring. In case that the crossover has not been performed, the offspring are identical to the parents. The following mutation operator is applied on each offspring. 6 integers are chosen randomly in the chromosome, each of which is mutated independently with the probability 50% by generating a new valid random value.

For each set of experiments (considering different number of CMR the transition function is composed of) 100 independent runs of the GA were performed. The evolution is terminated if a desired behavior of the candidate CA is observed (i.e. its chromosome obtained the maximal fitness value for a given problem) or if a given limit of generations is reached (this parameter is specific for the problems to be solved – see the next section).

A binary 2D uniform cellular automaton was used consisting of 24x24 cells. The evaluation of its behavior was performed within 16 developmental steps. The initial state of the CA is set as a fixed pattern (in this paper the initial state is not a subject of evolution, it is specified by the designer). The selection of the initial pattern and the way of calculating the fitness function depends on the problem to be solved and their description is covered in Section IV.

IV. Results and Discussion

For the purposes of this paper several problems were chosen (i.e. a specific behavior of the cellular automaton) for which the transition function has been designed by means of genetic algorithm in combination with the CMR-based representation. In this section, it will be shown that the evolution is able to design transition functions for non-trivial problems in CA using the 9-cell Moore neighborhood. In the simplest case of binary CA there are in total $2^9 = 512$ different transition rules and hence the search space contains 2^{512} possible transition functions if the conventional table-based representation is considered. As we demonstrated in [13], the success rate of evolving the tables for the replication problem and pattern development problem is substantially lower in most cases compared to advanced program-based transition function. For some problems the table-based representation even did not provide any working solution. In this section we propose results for the replication problem and pattern transformation problem using the CMR encoding of transition functions.

A. The Replication Problem

The goal of replication is to develop a copy of a given structure represented as a finite-size initial pattern in a finite number of development steps. The genetic algorithm was applied to design a transition function by means of which the CA develops so that there is a given number of copies of the initial structure after a finite number of development steps. One of the simplest techniques able to replicate an arbitrary structure is based on additive transition rules [2]. As we shown in [13], if such kind of transition function is discovered for a specific pattern used for training the CA, then the transition function is able to replicate different structures that were not considered during evolution. However, in [13], we were able to evolve only the aforementioned type of replication function.

```
best_fitness = 0  # fitness out of all development steps
const REPLICS = 2 # the minimal number of required replics

initialize the CA by the pattern to be replicated
FOR int step = 1 TO DEVEL_STEPS DO
{
   fitness = 0  # fitness in one development step
   replics_cnt = 0  # num. of replics found in a devel. step
   ca_step(cal, genome->prog);

   FOR row = 0 TO CA_HEIGHT - PATTERN_HEIGHT DO
   {
      FOR col = 0 TO CA_WIDTH - PATTERN_WIDTH DO
      {
         partial_fitness = 0  # fitness in specific part of CA
         FOR pr = 0 TO PATTERN_HEIGHT - 1 DO
            FOR pc = 0 TO PATTERN_WIDTH - 1 DO
               IF ca[row+pr][col+pc] == pattern[pr][pc] THEN
                  partial_fitness = partial_fitness + 1
         save the partial_fitness value
         IF found perfect pattern at position (row, col) THEN
            replics_cnt = replics_cnt + 1
      }
   }
   fit = sum of the REPLICS best saved partial fits
   # add a bonus if the solution produces more replics
   fit = fit + replics_cnt * PATTERN_HEIGHT * PATTERN_WIDTH

   # save the best fitness out of all development steps
   IF fitness > best_fitness THEN
      best_fitness = fitness
}

RETURN best_fitness
```

Fig. 4. The fitness function used for the replication problem (the same as in [13]). The pattern dimensions $PATTERN\_WIDTH$ and $PATTERN\_HEIGHT$ include a border consisting of a single line of inactive (zero-state) cells on each side because we required the replicated structures to be separated each other.

In this section, we demonstrate that some other replication processes can be found that (1) are not universal, i.e. the CA is able to replicate the pattern it was trained for but it fails if another pattern is specified, and (2) the replicated patterns can overlap. The fitness evaluation algorithm that was utilized during evolution is shown in Figure 4. In these experiments, we required to develop at least two instances of the initial pattern.

Statistical results related to the replication problem are summarized in Table I. The maximal number of generations was set to 500 thousands. If no solution is found within this limit, the evolution is stopped. For each experiment with a specific number of conditionally matching rules encoded in a chromosome 100 independent evolutionary runs were performed. The success rate, number of generations and rules of the evolved transition function (in the conventional table-based representation) was measured with respect to the number of CMR encoded in a chromosome during evolution. As the results show the success rate increases with increasing the number of conditionally matching rules. It indicates that there are more valid solutions in the search space that is represented by higher number of conditionally matching rules. An interesting phenomenon can be observed in the number of rules of the evolved transition functions in the table representation. The minimal number of rules tend to decrease slightly for

the increasing number of CMR. This observation is unusual because the more the CMR the more the table-based rules can be potentially generated. On the other hand, the maximal number of table-based rules exhibits rather an opposite trend. It indicates that the complexity of evolved transition functions transformed into the table representation rather depends on the general complexity of the CMR representation than on the number of CMR. Moreover, the lower number of CMR in general does not mean a reduction in complexity of the corresponding table-based transition function.

TABLE I. STATISTICAL RESULTS FOR THE REPLICATION PROBLEM CONSIDERING THE CMR-BASED APPROACH. THE NUMBER OF TABLE RULES REPRESENTS HOW MANY RULES COMPRISE THE CONVENTIONAL TABLE-BASED TRANSITION FUNCTION WHOSE APPLICATION MODIFIES THE STATE OF INVESTIGATED CELL.

Number of CMR	Success rate	Mean number of gen. (std. dev.)	Number of table rules: mean	min.	max.
08	27	252390 (131598)	265	232	288
10	39	173072 (124653)	267	197	298
12	43	196924 (135305)	265	205	294
14	64	163372 (110220)	264	188	302
16	70	168246 (92543)	270	193	337
18	70	135745 (101820)	267	195	330
20	82	125458 (95564)	270	218	330
22	89	139410 (92700)	260	195	319
24	86	152829 (99917)	260	196	341
26	94	146169 (101447)	258	203	320

In order to design a transition function for the replication problem, an initial (training) pattern was used as shown in Figure 5, part I. Figure 5 also shows one of the solutions that was found using genetic algorithm for the replication task. As evident, pattern I. is replicated after 8 steps and the CA produces more copies if the development continues. The replics are isolated each other – there is at least one line of zero-state cells between the neighboring rectangles delimiting the replicated structures. Part II. of Figure 5 shows another example of development using the same transition function. However, although the initial pattern is very simple and the CA is possible to produce some copies during development, the result is not the desired replication of the original pattern. More complex pattern is depicted in part III. of Figure 5. As the CA development shows, this transition function is not able to produce any copy of this structure because the shape of the original is destroyed. Hence the evolved replication function is not universal. In fact, the CA was trained for a specific pattern that was used for evaluating the candidate solutions. A specific feature of this result is that the process of development produces active cells only on the right of the initial pattern (i.e. there is no active development to the other sides). It seems that this solution is the simplest one for obtaining two replics in a limited number of steps. However, the obtained results contain solutions that replicate to one of the other sides which indicates that the evolution is able to find symmetric rules in order to create copies of the initial structure into the "empty" (zero-state) cells that are available inside the CA.

Another example of a successful result is shown in Figure 6. The same initial structure was used to train the CA during evolution. However, the replication process is different. In this case the direction of the replication is on the north-west side and the first complete replics arise after the fourth step. Although the shapes of the replics are isolated, their delimiting rectangles overlap by three cells (including the one-line of zero-state cells on each side of the shape). The experiments

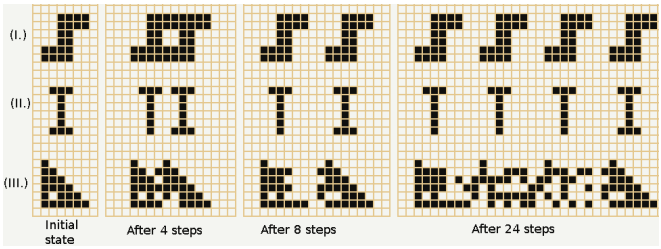


Fig. 5. Replication of some selected initial structures in a cellular automaton. This case represents an example of non-universal replicator, some structures are not replicated correctly.

showed that this kind of replication is much less common in the obtained results which indicate either a need of a more complex transition function or that this kind of transition function is rare in the search space. Similarly to the previous example, a result replicating into the opposite direction was also observed.

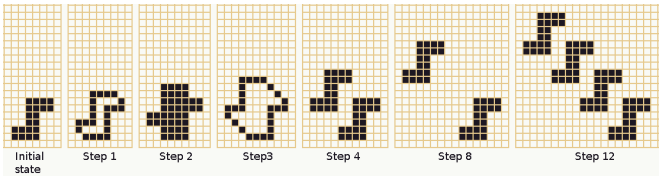


Fig. 6. Diagonal replication of an initial structures using the transition function from Figure 7

```
(0)!=0 | (1)!=1 | (2)==0 | (3)==0 | (4)<=0 |    *    | (6)==0 | (7)==0 | (8)!=1 | 0
(0)<=1 | (1)!=1 |    *   | (3)>=0 |    *    | (5)<=0 | (6)==0 |    *   | (8)==0 | 0
(0)<=0 |    *   |    *   |    *   | (4)==0 | (5)>=1 | (6)!=1 | (7)>=0 | (8)<=0 | 0
   *   |    *   | (2)>=0 | (3)>=0 | (4)<=0 |    *    |    *   |    *   | (8)>=1 | 1
(0)==0 | (1)==1 | (2)==1 |    *   | (4)<=0 |    *    |    *   | (7)>=1 | (8)!=0 | 0
   *   | (1)==0 | (2)<=1 | (3)==0 | (4)==0 | (5)==0 | (6)>=0 |    *   | (8)>=1 | 1
   *   | (1)>=0 | (2)>=0 |    *   |    *   | (5)!=0 | (6)!=1 | (7)>=1 | (8)>=1 | 0
(0)<=0 | (1)!=0 | (2)==0 | (3)==0 | (4)!=1 | (5)>=1 | (6)>=1 | (7)==1 |    *   | 0
(0)<=1 | (1)>=0 | (2)<=1 | (3)<=1 | (4)==1 | (5)>=0 | (6)<=1 | (7)<=1 | (8)!=0 | 0
(0)<=0 |    *   | (2)!=0 |    *   | (4)!=1 |    *   | (6)<=0 | (7)>=1 | (8)==0 | 1
   *   | (1)!=0 | (2)<=0 | (3)!=0 | (4)<=0 | (5)==0 | (6)!=1 | (7)!=0 |    *   | 0
   *   | (1)!=1 | (2)>=0 | (3)>=0 | (4)!=1 | (5)<=0 | (6)!=0 | (7)==1 | (8)>=0 | 0
```

Fig. 7. Evolved CMR-based transition function for the replication process from Figure 6 Each row represents a conditionally matching rule. The numbers in parentheses denote indices of cells in Moore neighborhood (according to Figure 1) whose states are evaluated within the conditions. Note that the cell indices are shown for convenience only, they are considered implicitly by the positions of conditions in the rules.

The proposed results show that various solutions exist for the replication of a given structure. In general, they may not be considered as universal replicators because some of them fail in replication of other structures. This feature is caused by the fact that a single specific pattern was considered during evolution of the transition function and, in fact, the CA is trained to this pattern only. It also means that the replication process itself may be specific for a given pattern. This issue might be interesting, for example, from a computational point of view. One of the hypotheses of this kind of research may be whether are there suitable structures whose replication could be considered as an efficient computation algorithm for a given task using CA in addition to currently known solutions (e.g. Tempesti Loops [14]).

B. The Pattern Transformation Problem

The objective of the pattern transformation problem is to find a transition function for a CA that is able to transform a specific pattern (represented by the initial state of the CA) into a given target pattern in a finite number of steps. For the purposes of this experiment, a counter-clockwise rotation by 90 degrees of the initial pattern will be considered. Note that this transform represents one of the problems whose solution was not successful using other CA design approaches (i.e. evolution of the transition function as a table or a program investigated in [13]).

Statistical results related to the pattern transformation problem are summarized in Table II. The maximal number of generations was set to 1 million. If no solution is found within this limit, the evolution is stopped. Similarly to the replication problem, the success rate tends to increase in most cases with the increasing number of CMR although the maximum observed success rate is significantly lower. However, the number of table rules exhibits an opposite trend compared to the replication experiments. For the increasing number of CMR both the minimal and maximal number of table rules tend to increase. It may indicate that the pattern transformation task is robust, i.e. the solution can be achieved in many different ways (both less and more complex) and the more CMR the more complex transition function can be found. This observation can also be confirmed by the resulting CA behavior for which (as shown later) different number of steps may be needed to transform the given pattern using variable transition functions.

TABLE II. STATISTICAL RESULTS FOR THE PATTERN TRANSFORMATION PROBLEM CONSIDERING THE CMR-BASED APPROACH. THE NUMBER OF TABLE RULES REPRESENT HOW MANY RULES COMPRISE THE CONVENTIONAL TABLE-BASED TRANSITION FUNCTION WHOSE APPLICATION MODIFIES THE STATE OF INVESTIGATED CELL.

Number of CMR	Success rate	Mean number of gen. (std. dev.)	Number of table rules: mean	min.	max.
08	21	406258 (486154)	115	58	178
10	34	353064 (425202)	119	67	250
12	27	299977 (319499)	140	100	214
14	45	250068 (229413)	147	88	221
16	47	229491 (213390)	147	87	318
18	48	224394 (227729)	152	96	209
20	60	180913 (204358)	163	124	234
22	64	158919 (156925)	169	81	273
24	45	163746 (168598)	187	99	338
26	53	208153 (213602)	175	110	270

The initial pattern used in our experiments is described in the upper-left part of Figure 8. The structure to be rotated is represented by a 10x10-cell shape including one line of zero-state cells on each side delimiting the given structure. The fitness evaluation is performed as follows. After each step of the CA a partial fitness is calculated as the number of cells in correct state in the 10x10-cell region. Note that the target state of each cell is determined according to the known pattern which represents the rotated initial 10x10-cell structure by 90 degrees counter-clockwise. The fitness value of a candidate transition function is the maximum from the partial fitness values. The pattern transformation is not a trivial task considering the fact that only local cell interactions are involved during the CA development. It means that the global behavior representing the process of rotation is an emergent feature of the CA.

Several perfect results have been obtained using the CMR-based approach. One of the results is shown in Figure 8. As evident, the initial pattern is precisely rotated after 13th step. Of course, no subsequent rotation will take place if the development continues because it represents another task for the CA that was not considered during evolution. In this case, the rotated pattern is destroyed during the next steps and the CA gets into a loop in which several states alternate periodically. The transition function that was evolved for the CA from Figure 8 is shown in Figure 9. The table-based representation of this transition function consists of 58 rules that change the state of the investigated cell which represents the most compact solution that was evolved in this paper.

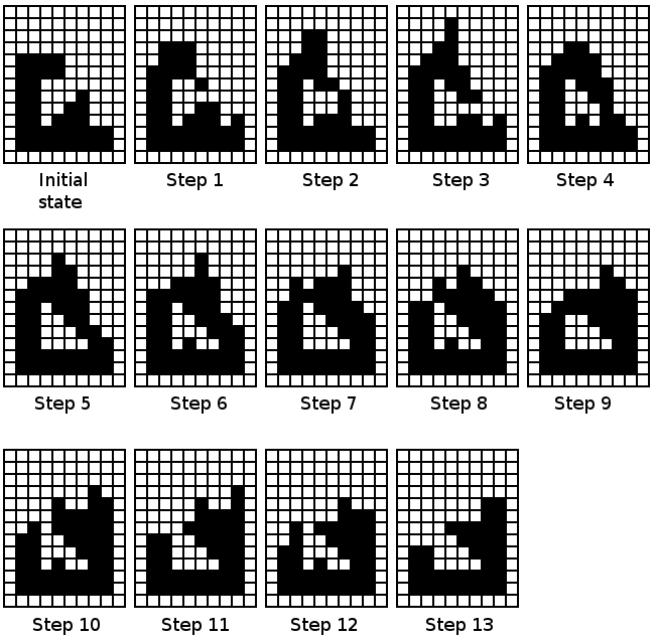


Fig. 8. Counter-clockwise rotation by 90 degrees in a cellular automaton. The pattern to be rotated is represented by the initial state. The rotation is performed in 13 steps using the transition function from Figure 9.

```
(0)!=1 |    *   | (2)!=1 | (3)<=0 | (4)==1 |    *   |    *   | (7)==1 | (8)>=0 | 0
(0)==1 | (1)!=0 | (2)==0 | (3)==0 | (4)<=1 | (5)==0 |    *   | (7)==1 | (8)<=0 | 1
(0)!=1 |    *   | (2)<=0 | (3)<=1 | (4)>=0 |    *   | (6)==1 | (7)>=1 | (8)==1 | 1
(0)!=1 | (1)==1 | (2)!=0 | (3)!=0 | (4)<=0 | (5)==1 | (6)!=0 |    *   |    *   | 1
(0)==1 | (1)>=0 | (2)==0 |    *   | (4)<=1 |    *   | (6)!=1 | (7)==0 | (8)==1 | 1
   *   | (1)!=1 | (2)<=1 |    *   | (4)<=1 | (5)==0 | (6)==1 | (7)>=1 | (8)==0 | 1
(0)<=0 |    *   | (2)!=0 | (3)>=1 | (4)>=1 | (5)<=1 | (6)==1 | (7)!=1 | (8)==0 | 0
(0)==1 | (1)<=1 | (2)==0 | (3)>=0 |    *   | (5)>=1 |    *   | (7)==0 | (8)<=0 | 1
```

Fig. 9. Evolved transition function for counter-clockwise rotation of an initial structure from Figure 8. Each row represents a conditionally matching rule. The numbers in parentheses denote indices of cells in Moore neighborhood (according to Figure 1) whose states are evaluated within the conditions. Note that the cell indices are shown for convenience only, they are considered implicitly by the positions of conditions in the rules.

Another perfect solution is shown in Figure 10 and the corresponding CMR-based transition function in Figure 11. This transformation shows a more intricate process; the initial pattern is precisely rotated after the 16th step. Moreover, if the development continues, another remarkable pattern emerges in step 30 - a triangular structure that was not explicitly considered during evolution. The "fate" of this triangle in this CA is not very good – it is going to disappear completely. However, the process during which it happens could be interesting. As

shown in Figure 10 (Step 50), a stair-like pattern is formed that successively removes the active cells at the hypotenuse of the original triangle. Although the triangle in step 30 consists of only 36 active cells, it takes in total 64 steps before it disappears.

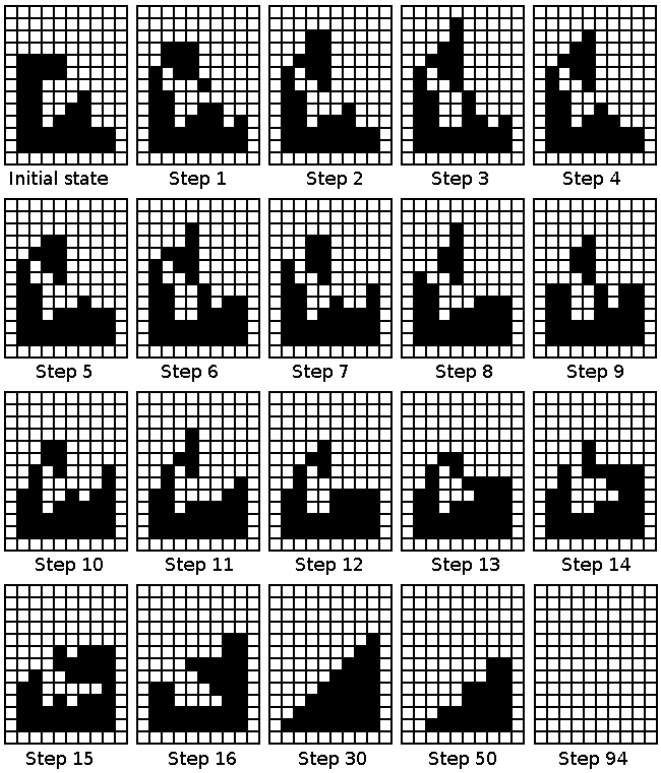


Fig. 10. Counter-clockwise rotation by 90 degrees in a cellular automaton according to the transition function from Figure 11

```
(0)>=1 | (1)!=0 |    *    |    *    | (4)<=0 | (5)==1 | (6)>=0 | (7)<=1 |    *    | 1
   *   |    *    | (2)>=1 | (3)==1 |    *    |    *   | (6)>=0 | (7)>=1 | (8)==0 | 0
(0)==1 | (1)>=0 | (2)==1 | (3)==0 |    *    | (5)>=0 | (6)==1 | (7)<=1 | (8)>=1 | 0
(0)>=1 | (1)>=0 |    *    | (3)>=0 |    *    | (5)==0 | (6)==0 |    *    | (8)==1 | 1
(0)!=0 | (1)!=1 | (2)>=0 | (3)==1 | (4)<=0 |    *    | (6)==1 | (7)!=1 | (8)>=0 | 0
   *   | (1)==0 | (2)!=1 | (3)<=0 | (4)!=0 | (5)>=0 | (6)<=1 | (7)<=1 | (8)>=0 | 0
(0)>=0 | (1)<=0 |    *    | (3)==0 |    *    | (5)==1 | (6)>=1 | (7)>=1 | (8)>=0 | 1
(0)!=1 |    *    |    *    | (3)>=0 |    *    | (5)<=0 | (6)!=0 | (7)==1 | (8)>=0 | 1
```

Fig. 11. Evolved transition function for counter-clockwise rotation of an initial structure from Figure 10. Each row represents a conditionally matching rule. The numbers in parentheses denote indices of cells in Moore neighborhood (according to Figure 1) whose states are evaluated within the conditions. Note that the cell indices are shown for convenience only, they are considered implicitly by the positions of conditions in the rules.

The same process can be observed for such triangles of different sizes. An example of a complete development is shown in Figure 12 for the triangle whose size (i.e. each its side) is represented by 4 active cells. This triangle is going to disappear after the 16th step which is interesting from a computational point of view. It can be observed that the number of steps for the triangle to disappear is equal to the square of the number of cells representing its size. For example, the triangle whose each side consists of 10 active cells needs $10^2 = 100$ steps to disappear. Note that this feature was not considered during evolution of the CA (the goal was only to perform rotation of the initial structure).

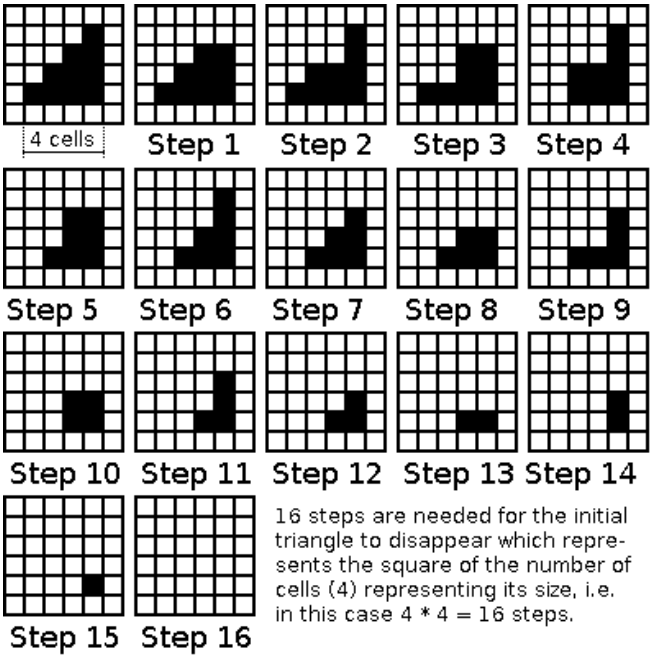


Fig. 12. Example of a process of disappearing a triangle whose number of steps represents the square of size of the triangle. The development is performed according to the transition function from Figure 11.

The solutions that were presented for the pattern rotation problem work in cellular automata of arbitrary size that is sufficient for representing the initial pattern. For example, if a 100x100-cell CA is used whose central region is initialized by the structure to be rotated, the transformation process will be performed correctly. However, other solutions have been observed whose functionality is limited only to the CA whose size corresponds to that considered during evolution. It means that some results can not be considered as general with respect to the CA size. The reason of this issue lies in the fact that only CA of finite sizes can be practically implemented in which a form of boundary conditions has to be applied. In our experiments, zero-boundary conditions were considered which in fact influences the CA development in a limited cellular space. In some cases the evolution utilized this feature and adapted the solution to the conditions of a finite CA size.

V. Conclusions

In this paper a method for representing transition functions for cellular automata has been presented. The proposed approach is based on introducing conditions into the transition rules that have to be satisfied in order to match the rule (i.e. to use it for determining the next state of a cell). One of the main features of this representation is that the number of elements needed to represent a transition function can be reduced in comparison with the conventional table-based representation. Yet, the possibility of specifying traditional transition rules (as in the table-based approach) is preserved which is suitable in situations when it is needed to determine a new cell state for a specific combination of states in the cellular neighborhood.

In the case study considering the replication problem, several solutions were designed using a genetic algorithm. It was determined that the CA is able to replicate a given

structure but failed in replicating some other structures that were not considered during evolutionary search of the transition function. It means that the evolution utilized specific features of the input pattern for which the replication rules were adapted. Therefore, other replication schemes may exist for different patterns which could be potentially useful, for example, for the purposes of performing computations using cellular automata.

The pattern transformation problem involved a counter-clockwise rotation by 90 degrees of a given structure in a cellular automaton. This task belongs to some previously studied problems whose solution failed using other approaches to the cellular automata design. In this paper, several perfect results were obtained demonstrating various solutions of the pattern transformation. It was determined that in some cases a computationally interesting behavior (specifically, calculation the square of a number representing the size of the input pattern) can be observed that was not explicitly considered during evolution.

This work was focused on binary cellular automata with Moore neighborhood. However, the results of our subsequent experiments indicate that the proposed approach can be applied for the design of cellular automata working with higher number of states. Therefore, our next research will be devoted to the design of complex cellular automata for which the conventional approaches do not provide satisfactory results. In particular, more non-trivial patterns will be studied in the replication problem (including self-replicating loops) and the resulting cellular automata will be also analysed with respect to their computational properties. Another interesting research area could be the CMR approach itself in which each CMR might be evaluated in order to determine its contribution to achieve a given CA behavior. A study of these features might enable to optimize the evolvability of the CMR representation.

ACKNOWLEDGMENT

This work was supported by the Czech Science Foundation project P103/10/1517 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[2] S. Wolfram, *A New Kind of Science*. Champaign IL: Wolfram Media, 2002.

[3] M. Sipper, *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194*. Berlin: Springer-Verlag, 1997.

[4] J. F. Miller, "Evolving developmental programs for adaptation, morphogenesis and self-repair," in *Advances in Artificial Life. 7th European Conference on Artificial Life, Lecture Notes in Artificial Intelligence, volume 2801*. Dortmund DE: Springer, 2003, pp. 256–265.

[5] P. C. Haddow and G. Tufte, "Bridging the genotype–phenotype mapping for digital FPGAs," in *Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware*. Los Alamitos, CA, US: IEEE Computer Society, 2001, pp. 109–115.

[6] G. Tufte and P. C. Haddow, "Towards development on a silicon-based cellular computing machine," *Natural Computing*, vol. 4, no. 4, pp. 387–416, 2005.

[7] T. Kowaliw, P. Grogono, and N. Kharma, "Bluenome: A novel developmental model of artificial morphogenesis," in *Proc. of the Genetic and Evolutionary Computation Conference, GECCO 2004, Lecture Notes in Computer Science, part I., volume 3102*. Springer-Verlag, 2004, pp. 93–104.

[8] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[9] F. Dellaert and R. Beer, "Toward an evolvable model of development for autonomous agent synthesis," in *Proc. of the 4th International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*. MIT Press, 1994, pp. 246–257.

[10] ——, "A developmental model for the evolution of complete autonomous agents," in *Proc. of the 4th International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press-Bradford Books, 1996, pp. 393–401.

[11] Y. Guo, G. Poulton, G. James, P. Valencia, V. Gerasimov, and J. Li, "Designing stable structures in a multi-agent self-assembly system," in *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, 2004, pp. 405–408.

[12] Y. Kayama, "Network view of binary cellular automata," in *Cellular Automata for Research and Industry*, ser. Lecture Notes in Computer Science Volume 7495. Springer Verlag, 2012, pp. 224–233.

[13] M. Bidlo and Z. Vasicek, "Evolution of cellular automata using instruction-based approach," in *2012 IEEE World Congress on Computational Intelligence*. IEEE Computer Society, 2012, pp. 1060–1067.

[14] G. Tempesti, "A new self-reproducing cellular automaton capable of construction and computation," in *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life*, ser. Lecture Notes in Artificial Intelligence, vol. 929. Springer Verlag, 1995, pp. 555–563.

Appendix III

Evolving Multiplication as Emergent Behavior in Cellular Automata Using Conditionally Matching Rules

BIDLO Michal

In: *2014 IEEE Congress on Evolutionary Computation (CEC 2014)*. Beijing: IEEE Computational Intelligence Society, 2014, pp. 2732-2739. ISBN 978-1-4799-1488-3.

Conference CORE rank in the year of publication: B

2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Evolving Multiplication as Emergent Behavior in Cellular Automata Using Conditionally Matching Rules

Michal Bidlo

***Abstract*— In this paper a special representation technique called conditionally matching rules will be applied in order to design computational processes in uniform cellular automata. The goal is to verify abilities of this approach in combination with genetic algorithm on the problem of disigning various cellular automata that exhibit a given computational process. The principle of a computational process in a cellular automaton is to interpret some cells as input bits and some (possibly other) cells as output bits (i.e. the result of the computation). The genetic algorithm is applied to find a suitable transition function of a cellular automaton according to which the given computation could be observed during its development for all the possible binary combinations stored in the input cells. Both the input values and the result is represented by state values of cells. The input of the computation will be represented by the initial state of the cellular automaton. After a finite number of development steps the cells representing the output bits are expected to contain the result of the computation. A set of experiments will be performed considering various setups of the evolutionary system and arrangements of the target computation. It will be shown that non-trivial computations can be realized in a uniform two-dimensional cellular array.**

I. Introduction

Cellular automata (CA) represent a biologically inspired dynamical system with a discrete time and space. Cells represent basic computational elements of a CA. At a given moment each cell possesses a value representing its state from a finite set of states. The cell states can be considered as data (information) units processed by the cellular automaton. The concept of cellular automata was originally invented by Ulam and von Neumann in 1966 [1] for studying the behavior of complex systems.

A two-dimensional (2D) cellular automaton consists of a regular grid of cells that are arranged into a regular matrix (mesh). In each (discrete) developmental step of the CA the states of all the cell are updated synchronously in parallel according to a local transition function. The next state of a given cell depends on the combination of states in its neighborhood (including the cell itself). A sequence of updating the cell states during discrete time steps represents development of the cellular automaton.

For the purposes of this paper the following concept of the cellular automata will be considered. The cellular neighborhood of each cell is represented by a 9-tuple (3x3 cells) consisting of the investigated (central) cell and its immediate neighbors in the horizontal, vertical and diagonal directions. Since only finite-size cellular automata can be practically implemented, boundary conditions will be defined for the cells at the border of the cellular mesh. In this paper zero boundary conditions will be applied which means that non-existing neighbors of the border cells are considered as notional cells in a permanent state 0.

Conventionally the local transition function is represented by a table that specifies the next state of a cell for all the possible combinations of states in its neighborhood. However, if the number of cell states or the size of the cellular neighborhood increases, then the number of such combinations grows exponentially and thus the representation and design of the transition function becomes a challenging task.

In order to overcome this issue, a new technique for representing the transition functions was introduced by Bidlo et al. and called as Conditionally Matching Rules (CMR) [2]. This approach is fundamentally inspired by the conventional table-based representation. It means that the CMR encoding allows to specify the transition rules as usual in the table-based approach but, in addition to that, more general rules can be formulated whose interpretation covers several common rules in a single CMR. In particular, a conditionally matching rule consists of a conditional part and a next state. The conditional part encodes a series of pairs — a condition function and a state value — whose number corresponds to the number of cells in the cellular neighborhood. The structure of a CMR is illustrated in the upper part of Figure 1. A local transition function of a CA consists of a finite sequence of conditionally matching rules. The process of determining the next state of a cell using the CMR-based transition function is the following. The CMRs are evaluated sequentially one after another until a CMR matches the states in the cellular neighborhood. In order to determine a CMR match, each rule of its conditional part is evaluated with respect to the corresponding cell state in the neighborhood (see Figure 1). If all the conditions are satisfied, then the next state from the matching CMR represents the result of the transition function (i.e. the new state of the investigated cell) and none of the remaining CMRs in the sequence needs to be evaluated. If none of the CMRs representing the transition function matches, then the cell keeps its current state. The experiments showed that if the CMR approach is utilized to represent transition functions, then more complex cellular automata can be effectively evolved in comparison with the traditional representation [2]. Hence the advanced features and abilities of the CMR representation are worth the next investigation.

In addition to a wide range of applications utilizing cellular automata to solve some specific tasks (e.g. modeling complex biological and physical systems, artificial live, random number generation and many others [3]), cellular automata

Michal Bidlo is with the Faculty of Information Technology, Brno University of Technology, IT4Innovations Centre of Excellence, Božetěchova 2, 61266 Brno, Czech Republic, email: bidlom@fit.vutbr.cz.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

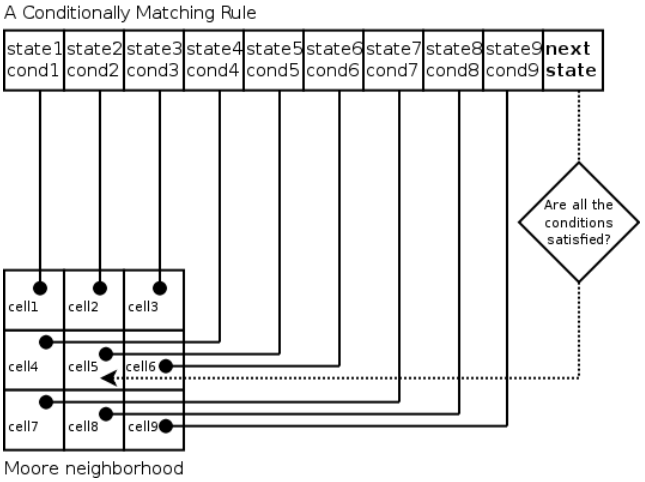


Fig. 1. Structure and interpretation of a conditionally matching rule

also represent a platform to perform computations. Various concepts of computation performed in cellular automata have been studied both theoretically and practically (i.e. using real implementations in FPGAs or as application-specific circuits [4][5]). The importance of undertaking such kind of research is motivated especially by the fact that cellular automata represent (in many cases) homogeneous and massive parallel computing platform with typically local interactions of cells. The issue of homogeneity may be important in a process of designing large systems whose elements (computational units – cells), for example, perform a given function that, in cooperation with each other, realizes a specific (emergent) behavior. An advantage of such system may be its scalability (it is easy to connect additional cells with the same function) or a possibility of repair in case of a cell failure. Advanced concepts may consider specializations of different cells during the system functioning. This idea is mostly inspired by multicellular (biological) systems in which a target organism can grow during its life and the cells change their kind with respect to a location in the organism or on the basis of external conditions. Some studies and applications of these issues (in a more general conception referred as computation development) were published in [6]. Computational universality of cellular automata was proven for the first time by their inventor John von Neumann in [1]. His CA worked with 29 cell states which was later reduced to 8 states by Codd [7]. Lindgren and Nordahl demonstrated that even 1D cellular automaton can be utilized as universal computing platform (i.e. a platform that is able to simulate the computation using Turing machine). They proposed a proof of universality for the 7-state 1D CA with 3-cell neighborhood and 4-state 1D CA with 5-cell neighborhood [8]. Sipper showed that 2D binary non-uniform cellular automata are able to perform computations by demonstrating how to realize computationally complete set of logic functions and their interconnection [4]. Uniform 2D binary cellular automaton was demonstrated to be a computationally universal platform using the popular Conway's Game of Life rules [9]. The idea was to utilize some "living" cell structures like glider guns, gliders, periodic patterns etc. to simulate the computational process of Turing machine. These structures represent a means for implementing basic logic gates, synchronization mechanism and memory elements which represent fundamental components of a universal computer. Another interesting computationally universal 2D CA is represented by the rules of Langton's ant [10]. His CA represents a model with simple transition rules that is able to exhibit complex emergent behavior. A proof of universality of Langton's ant was proposed in [11]. Other (not necessarily universal) cellular automata able to perform specific compatational tasks were also published (e.g. Langton's loop implementing self-replication [12], Tempesti loop which added an ability of construction [13] and others).

The goal of this paper is to show that various CA setups together with some appropriate evolutionary system setups are able to design transition functions allowing us to perform a specific computational task in the CA. In particular, we propose two sets of condition functions for the CMR in combination with various input and output cell arrangements in order to design CA whose development exhibits 2x2-bit multiplication. The objective of studying various CMR setups is to determine how the different sets of condition functions influence the ability of the evolutionary algorithm to find a solution of a given task in cellular automata. The experiments will be evaluated with respect to the success rate and computational effort of the evolutionary design process. Features and abilities of the obtained results will be discussed with respect to a future research.

II. Setup of Conditionally Matching Rule

The original concept of the conditionally matching rules introduced in [2] considered ordinary relational operators equal to ($==$), not equal to ($!=$), greater or equal than ($>=$), less or equal than ($<=$) and a don't care mask ($\star$) as the set of conditions. In case of binary cellular automata, where the state of a cell can be either 0 or 1, each condition in a CMR actually represent a function with two binary inputs (where one bit represents the state of a cell from the cellular neighborhood, the other is a state value specified in the conditional part). The function calculates a single-bit output indicating whether the given part of the CMR matches the corresponding cell state. Table I shows results of the aforementioned functions for all their possible input values.

Since there are 4 possible input combinations, $2^4 = 16$ different functions exist in total which could be used as condition functions in CMRs. Our previous experiments showed that it is not suitable to consider all the 16 functions because the space of the transition functions becomes very huge and the problem of finding a specific behavior of the CA represents a challenging task. Therefore, the selection of a subset of condition functions represents a reasonable solution. However, the main question is how to select this subset for the CMR in order to allow efficient design of transition functions for cellular automata. In [2] it was demonstrated that the subset of functions summarized in

TABLE I

THE TRUTH TABLE FOR THE ORIGINAL CONDITION FUNCTIONS $==$, $!=$, $>=$, $<=$ AND A DON'T CARE $\star$. IF THE RESULT OF A CONDITION FUNCTION IS 1, THEN THE APPROPRIATE CELL STATE MATCHES THE STATE SPECIFIED IN THE CONDITIONAL PART WITH RESPECT TO THIS FUNCTION. S_{CELL} DENOTES THE STATE OF A CELL IN THE CELLULAR NEIGHBORHOOD, S_{CMR} REPRESENTS THE STATE SPECIFIED WITHIN A CONDITION IN THE CMR.

Inputs to condition function		**Condition function**				
S_{CELL}	S_{CMR}	$==$	$!=$	$\leq$	$\geq$	$\star$
0	0	1	0	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	1	1	1

Table I represents one of the possible choices in order to achieve non-trivial behavior in binary 2D cellular automata. This subset was chosen experimentally on the basis of analyzing the target CA behavior (the replication and pattern transformation task).

In order to determine whether a more suitable subset of condition functions exists, the following approach was considered for the experiments presented in this paper. There are in total $2^{16} = 65536$ different subsets considering the complete set of 16 condition functions. It is possible to perform exhaustive search through all the subsets and evaluate the number of different functions that can be realized using the proposed CMR-based representation. For the purposes of this paper, the problem of calculating single-output binary functions with 9 inputs was considered in order to evaluate each subset of the condition functions. In fact, this setup is identical to exploring transition functions for 2D uniform binary cellular automata with 9-cell Moore neighborhood. Since it is impossible to perform in a reasonable time the exhaustive search of all tle possible 9-input functions, the evaluation was performed by generating 1 billion random samples of CMR sequences (considering the number of CMRs from 1 to 8). This experiment showed that one of the highest numbers of different functions can be generated using the subset of condition functions summarized in Table II. As evident, the resulting subset of condition functions contains, in addition to the relational operators $<=$ and $>=$, the identity function of the cell state (let us denote it as $id(S_{CELL})$ where S_{CELL} represents the cell state), the negation of S_{CELL} ($not(S_{CELL})$) and does not contain don't care mask ($\star$). Note that in this experiment no specific function (or CA behavior) was required, only the number of various functions was observed. The goal of this experiment was to identify a subset of condition functions for the CMR representation that would be potentially able to solve as wide set of tasks as possible.

Figure 2 shows an example of transition function represented by three conditionally matching rules. The CMRs utilize the condition functions from Table II. In order to determine the next state of the investigated cell (denoted by the thick rectangle in Figure 2), the CMRs are evaluated

TABLE II

THE TRUTH TABLE FOR A NEW SET OF CONDITION FUNCTIONS THAT MAY POTENTIALLY BE ABLE TO GENERATE A WIDER RANGE OF FUNCTIONS USING THE CMR REPRESENTATION.

Inputs to condition function		**Condition function**			
S_{CELL}	S_{CMR}	$id(S_{CELL})$	$not(S_{CELL})$	$\leq$	$\geq$
0	0	0	1	1	1
0	1	0	1	1	0
1	0	1	0	0	1
1	1	1	0	1	1

sequentially one after another in order to find a CMR that matches the state of the cellular neighborhood. For the CMR #1 it can be seen that condition (2) — the identity function — does not match because the state of cell (2) in the neighborhood has state 0. Therefore, CMR #1 can not be applied to determine the next state. Considering the CMR #2, it can be verified that this CMR fulfills all its conditions with respect to the states in the cellular neighborhood (the rule (1) satisfies the condition $S_{CELL} >= 1$ because $S_{CELL(1)} = 1$, the rule (2) after its evaluation matches the state of cell (2) because $not(S_{CELL(2)}) = not(0) = 1$ and so on). Therefore, the CMR #2 will be applied to determine the next state of the investigated cell, i.e. its new state will be 0.

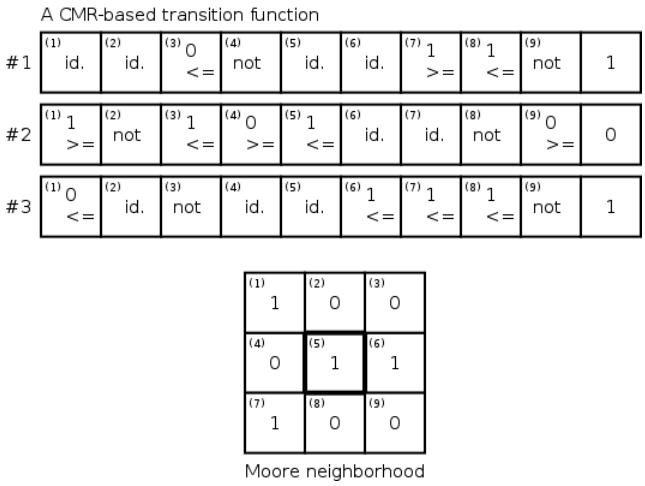


Fig. 2. Example of a transition function represented by conditionally matching rules. Note that the identity function (id) and negation (not) do not need any state value in the conditional part of the CMR because the decision of matching their part of CMR is based only on evaluating the appropriate cell state in the cellular neighborhood. The thick rectangle denotes the cell for which the new state ought to be calculated.

III. EVOLUTIONARY SYSTEM SETUP

Genetic algorithm (GA) was utilized for the evolution of CMR-based transition functions in order to realize 2x2-bit multiplication in 2D uniform binary cellular automata.

The population consists of 16 individuals (chromosomes) that are initialized randomly at the beginning of the evolutionary process. Each chromosome represents a candidate transition function represented as a finite sequence of CMRs.

The structure of each CMR is identical to that shown in the top part of Figure 1. Each CMR is encoded as a finite sequence of integers representing the conditional parts (i.e. the states and condition functions) and the next state.

The fitness function implements the following multiplication scheme in the cellular automata for evaluating the chromosomes. A binary input test vector (representing the operands to be multiplied) is generated into the given cells (let's call them the input cells) as the initial state of the CA. For 2x2-bit multiplication there are 4 input cells, i.e. two 2-bit operands. All the other cells are initialized by the state 0. Now the CA performs 16 development steps according to the transition function encoded in the chromosome after which the result of multiplication is verified as a sequence of states of the given (output) cells. For 2x2-bit multiplication there are 4 output cells, i.e. a 4-bit product. The fitness value is increased by one for every correct bit of the result with respect to the input vector. The evaluation is performed for all the possible input test vectors. For 2x2-bit multiplication there are $n = 2^4 = 16$ input test vectors and $p = 4$ bits of the product. Therefore, the maximal fitness for the 2x2-bit multiplication $F_{mult} = n \times p = 16 \times 4 = 64$. Moreover, we required the cells of the result to keep the states representing the product during the subsequent CA development. Hence the cell states representing the product are compared to the values after performing one more step of the CA for each of the input test vectors. If all the output cells keep their resulting states, the fitness is increased by one. Note that the input values are usually modified during the CA development (there is no requirement to keep them within the input cells). The maximal value of the complete fitness can be expressed as $F_{max} = F_{mult} + n = 64 + 16 = 80$.

Each step of the evolution is performed by generating offspring from parent chromosomes using a mutation operator until an entire new population is created. The parent chromosome is selected using tournament operator out of T chromosomes randomly chosen from the actual population, i.e. the fittest individual from the group of T chromosomes becomes the parent. The parameter T is referred to as the base of the tournament selection operator. The parent undergoes mutation as follows. A random integer M in the range from 1 to 4 is generated. Then M random positions within the parent chromosome are selected. The offspring is created by replacing the actual genes at these positions by new randomly generated values. No crossover operator is applied.

In order to explore the possibilities of realizing the proposed multiplication scheme, several sets of experiments were performed considering various setups. The two sets of condition functions presented in Table I and II were investigated within the evolutionary process. The GA was parametrized by the base of the tournament operator for $T = 2, 4, 6, 8$. The number of CMRs ($\#CMRs$) of the transition function was considered for $\#CMRs = 6, 8, 10, 12$. For each combination of these parameters, several setups of the input and output cells were considered as shown in Figure 3. The cellular automaton consisting of 10x10 cells was used for the evaluation of the chromosomes. The structure containing the input and output cells is located in the middle of the CA. The reason for choosing larger CA is not to strictly limit the computation process by the boundary conditions (required for finite-size CAs).

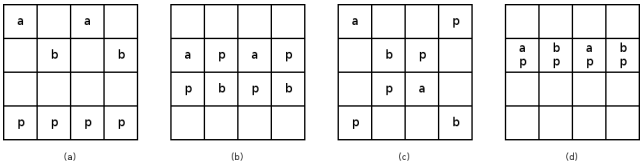


Fig. 3. The setups for input and output cell arrangements in cellular automata: (a) separated, (b) alternating, (c) diagonal, (d) shared. In each of the setups, a and b denote the input cells whose states represent values for the 2-bit operands, p denote the output cells in which the resulting product is expected after the CA development. Note that in the shared setup (d) the result is expected in the same cells as the input operands.

For each set of experiments (specified by the aforementioned parameters and cell setups) 100 independent evolutionary runs were performed. The evolution is terminated if the desired behavior of the candidate CA is observed (i.e. a chromosome with the maximal fitness value is found) or if a limit of 100 thousands generations is reached.

IV. Experimental Results

In this section we propose an overview of the experimental results obtained from the evolutionary system described in Section III. For each set of experiments the success rate and computational effort (measured as the number of generations of the GA needed to find a solution) were evaluated. The experiments showed that it is possible to realize the given computational task in uniform 2D binary cellular automata. We also determined that the required behavior discovered by the evolution may not be limited to a specific CA size.

The results of evolution using the original condition functions from Table I are shown in Table IIIa. The results of the application of the new set of condition functions from Table II are summarized in TableIIIb. In each set of experiments the success rate and computational effort of the evolutionary process was investigated with respect to the base of tournament selection (T) and the number of conditionally matching rules ($\#CMRs$) of the transition function. As the results show for both sets of the condition functions, the success rate tends to increase with increasing the $\#CMRs$ which indicates that although a larger search space is needed to explore (with potentially higher amount of target solutions), then the evolution is able to explore it effectively and in many cases even with less computational effort (expressed by the number of generations $Avg.\#gen$). Similar trend can also be observed for the increasing the base of tournament selection T. This parameter actually increases the selection pressure during evolution (the higher the T the higher the selection pressure) which means that the individuals with higher fitness are able to generate offspring chromosomes towards the target solutions within the search

TABLE III

STATISTICAL RESULTS OF THE EVOLUTIONARY EXPERIMENTS PERFORMED FOR VARIOUS NUMBERS OF CONDITIONALLY MATCHING RULES ($\#CMRs$) AND VARIOUS BASE VALUES OF THE TOURNAMENT SELECTION IN GA (T). THE SUCCESS RATE (**Succ.**) AND AVERAGE NUMBER OF GENERATIONS (**Avg.#gen**) NEEDED TO FIND A WORKING SOLUTION WERE MEASURED. **Std. dev.** DENOTE THE STANDARD DEVIATION CALCULATED FROM THE NUMBER OF GENERATIONS OF THE SUCCESSFUL EVOLUTIONARY RUNS.

(a)

Tour. base (T)	Condition functions: ==, !=, <=, >=, *															
	2			4			6			8						
	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.				
Cell configuration: shared																
#CMRs																
6	18	43779	(26631)	36	39643	(25500)	41	36380	(30311)	39	25566	(21603)				
8	41	44379	(25528)	52	35861	(32142)	55	34048	(26427)	50	40202	(30905)				
10	49	44632	(29822)	68	32456	(27002)	64	36913	(29753)	67	26418	(23922)				
12	65	32517	(26831)	69	33163	(27151)	64	33181	(28909)	65	29905	(27414)				
Cell configuration: diagonal																
6	0	0	(0)	2	43630	(17435)	0	0	(0)	1	40127	(0)				
8	12	55578	(26011)	6	44625	(15022)	10	46828	(22333)	7	35750	(14516)				
10	11	41269	(24952)	10	52182	(36465)	7	43203	(19738)	8	55822	(32119)				
12	11	30680	(16933)	8	34804	(23753)	9	43803	(29622)	7	38076	(31234)				
Cell configuration: alternating																
6	29	35659	(20203)	42	34159	(24357)	29	33959	(21370)	35	33455	(26175)				
8	56	35679	(22374)	59	34394	(27298)	59	28952	(23590)	55	37624	(27978)				
10	65	31004	(26140)	59	40752	(27497)	56	34636	(30389)	55	35953	(26045)				
12	51	36314	(30438)	62	30895	(27441)	56	31627	(25868)	50	34129	(29677)				
Cell configuration: separated																
6	1	5397	(0)	0	0	(0)	1	997	(0)	0	0	(0)				
8	2	18768	(9747)	2	37830	(33201)	3	15357	(11051)	3	45903	(37070)				
10	1	6800	(0)	3	19407	(4201)	3	44484	(25254)	7	49739	(31078)				
12	6	30386	(31832)	5	47820	(26963)	9	16859	(11379)	6	30998	(18015)				

(b)

Tour. base (T)	Condition functions: id, not, <=, >=															
	2			4			6			8						
	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.	Succ.	Avg. #gen.	Std. dev.				
Cell configuration: shared																
#CMRs																
6	13	46210	(23413)	29	46703	(31028)	29	24739	(25926)	29	34025	(31781)				
8	28	46675	(28353)	41	36787	(32183)	43	32558	(31288)	47	30162	(29529)				
10	54	38007	(29083)	67	27790	(26676)	56	31591	(26245)	50	29010	(28089)				
12	63	39791	(30414)	67	32594	(30283)	64	23458	(25731)	59	27177	(24981)				
Cell configuration: diagonal																
6	0	0	(0)	0	0	(0)	0	0	(0)	1	22194	(0)				
8	4	34011	(11507)	2	43199	(6633)	2	70546	(14364)	4	48787	(31622)				
10	8	54965	(19047)	13	38688	(16038)	6	29302	(21823)	4	16412	(4297)				
12	11	35774	(25174)	4	44549	(28406)	4	44685	(24314)	6	44254	(26395)				
Cell configuration: alternating																
6	21	47586	(27487)	19	46461	(25673)	14	43719	(28536)	19	32489	(32935)				
8	48	44060	(28490)	57	45571	(30263)	55	40997	(27071)	52	32761	(29059)				
10	55	39137	(27303)	62	33950	(23033)	59	31891	(24961)	49	45105	(28764)				
12	50	29540	(25917)	58	37293	(26802)	59	30876	(27665)	47	31773	(29253)				
Cell configuration: separated																
6	0	0	(0)	0	0	(0)	0	0	(0)	0	0	(0)				
8	1	50943	(0)	4	21267	(18896)	2	41303	(25965)	1	6445	(0)				
10	2	13244	(11196)	1	54157	(0)	3	15650	(6645)	5	36718	(30015)				
12	2	87193	(6309)	4	21590	(16344)	12	39859	(30182)	7	31446	(33834)				

space. However, the comparison of the results in Table III shows that in most cases the success rate is lower for the new set of condition functions from part (b) in comparison with the original set – part (a). This result is surprising because the new function set was evaluated to be able to realize a wide range of functions as described in Section II. Therefore, it is not possible to conclude that this function set would be generally more efficient than other sets. The lower success rate may be caused by the fact that the new function set in combination with the CMR approach represents a search space in which the target functions (regarding the multiplication task) are rather rare in comparison with the original set of condition functions. The optimal set of condition functions is thus evidently specific for a particular problem. Therefore, a more suitable function set may exist for solving the multiplication problem in cellular automata. However, to discover such optimal set still remains an open question.

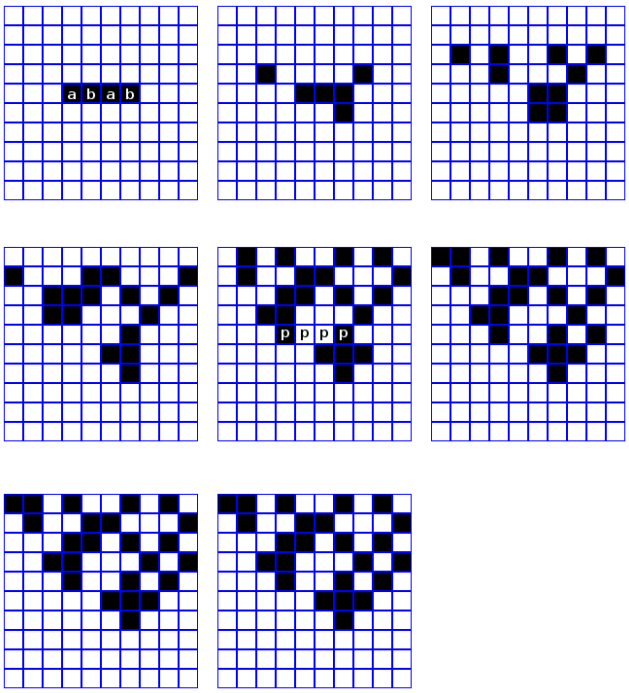


Fig. 4. Example of an evolved CA development performing the multiplication $3 \times 3 = 9$. White cell in the CA represents the state 0, black cell represents the state 1. The input operands as well as the resulting product are considered in direct binary representation using the cell states (the operands $a = b = (11)_{bin} = (3)_{dec}$, the product $p = (1001)_{bin} = (9)_{dec}$). Both the operand bits and product bits are interpreted from MSB to LSB as from left to right.

<=1	<=0	<=1	<=1	<=1	id	not	id	id	1
id	id	id	id	not	>=0	<=0	id	id	0
<=0	<=1	id	<=0	not	<=1	<=1	not	not	0
>=1	<=1	not	id	id	>=1	id	id	>=0	1
not	not	<=0	<=0	<=0	<=0	id	id	id	1
<=0	id	<=1	>=0	id	<=1	<=0	<=0	>=1	1

Fig. 5. Example of a CMR-based transition function discovered by GA using the function set from Table II. This transition function realizes multiplication in the CA from Figure 4. One row in this figure represents a CMR. The converted table-based representation consists of 31 conventional rules.

Nonetheless, both the proposed sets of condition functions allowed us to find some working solutions for different input and output cell configurations illustrated in Figure 3. As evident from Table III, the success rate is highly influenced by the cell setup. It can be seen that the highest success rates (greater than 60%) were achieved using the shared and alternating cell setups. The other setups (diagonal and separated) exhibit substantially lower success rates. The following reasons may be given for this result. If the input and output cells are close to each other, then the CA needs less steps to produce the result (i.e. the input values can be processed rather locally). On the other hand, if the inputs are distributed in a larger area or if the result is expected away from the inputs, then a more complex transition function is needed in order to process the inputs and transfer the resulting values to the output cells (the CA needs more steps to produce the result).

Figure 4 shows an example of evolved CA development performing the multiplication of two 2-bit operands. The appropriate CMR-based transition function discovered by GA is shown in Figure 5. The CA was initialized by the operand values within the input cells according to the shared cell setup shown in Figure 3d. In this case the CA needs 4 steps to produce the final product. As evident from Figure 4, the state of the output cells become stable during the subsequent CA development as required within the evolutionary process. In addition to the fact that the CA provides correct results for all the possible input operands, its development seems chaotic. However, an interesting emergent behavior can be observed when a larger CA is considered. For example, a final state of a 40x40-cell CA after the 30th development step is shown in Figure 6 with the input/output cells marked by p. The same task (i.e. to calculate the expression 3×3) was considered according to the setup from Figure 3d and the evolved transition function from Figure 5. Interestingly, the global CA behavior is similar to that produced by some typical rules of the elementary 1D cellular automata (for example, using the rules 22, 122 or 126 according to Wolfram code [3]). In our case, the pattern in Figure 6 was generated within a 2D CA by a successive "growth" of the cells upwards, developing a structure very similar to Sierpinski triangle. Although this structure can be obtained in a 1D CA using relatively simple rules from an initial seed, the transition functions obtained in this paper do not seem to be trivial. Similar emergent behavior can be observed within the development from a random initial state in a CA with enough area of 0-state cells available above the randomly initialized cells (see Figure 7). One of the open questions related to this issue is whether such emergent behavior could provide us with some advantages to perform (advanced) CA-based computation (e.g. using a specific operand encoding or interpretation of the CA development).

Another example of a successful multiplication in CA is illustrated in Figure 8 using the diagonal cell setup according to Figure 3c. This CA was discovered using the set of condition functions from Table I, the evolved transition function is shown in Figure 9. In this case the CA needs 10 steps to produce the result. After stabilizing the states of the output cells, the CA development exhibits an emergent pattern expanding from top to bottom and from left to right. In addition, a simple vertical line of state-1 cells grows upwards. We have determined that the number of those vertical lines depends on the number of 1's in the resulting product. This growing structure is very important in this particular CA because it contains the crucial states of the result. An example of the multiplication $3 \times 3 = 9$ is shown in Figure 10. As evident there are two 1s in the resulting product ($(9)_{dec} = (1001)_{bin}$) so that two state-1 lines grow upwards. The emerging pattern can be observed in the bottom part of Figure 10.

V. Conclusions

This paper presented a continuation of research regarding the evolutionary design of 2D uniform cellular automata

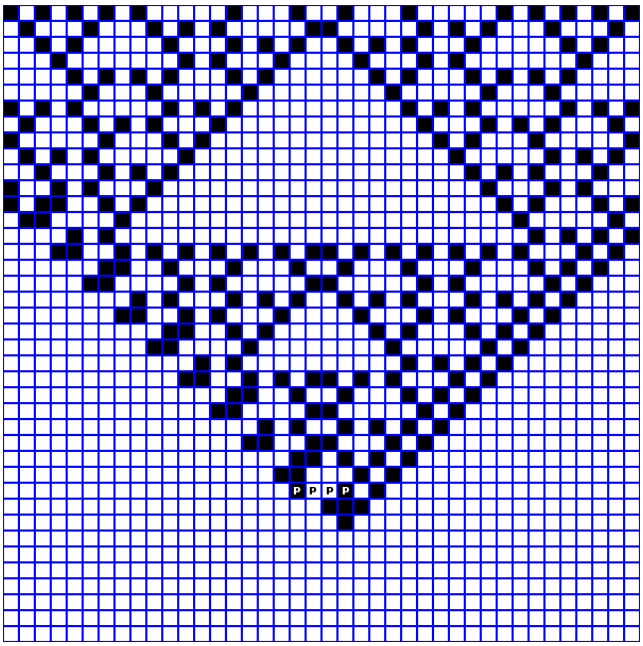


Fig. 6. The state of the 40x40-cell CA after the 30th development step performing the same multiplication task as shown in Figure 4 according to the transition function from Figure 5. The output cells containing the resulting product are denoted by p. These four cells were also initialized before the CA development by the values of the input operands. This is a final stable state of the CA that does not change anymore.

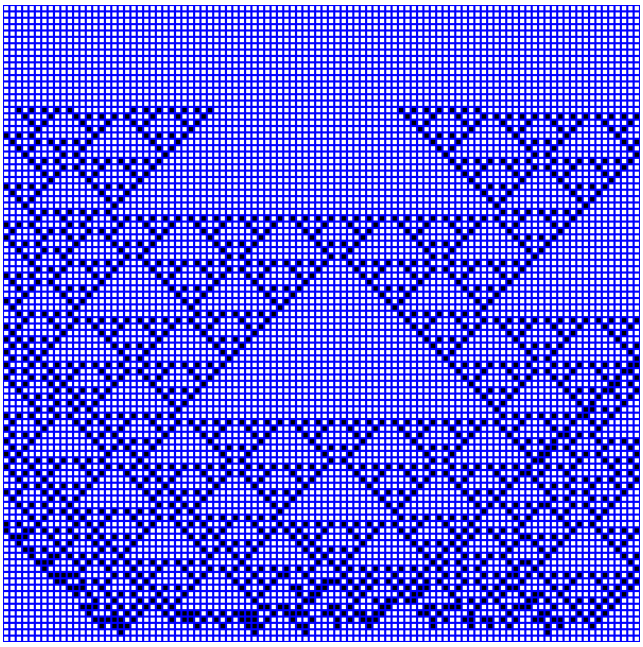


Fig. 7. An emergent pattern developed by 2D CA from some randomly initialized cells at the bottom of the cellular array. The CA develops according to the evolved transition function from Figure 5.

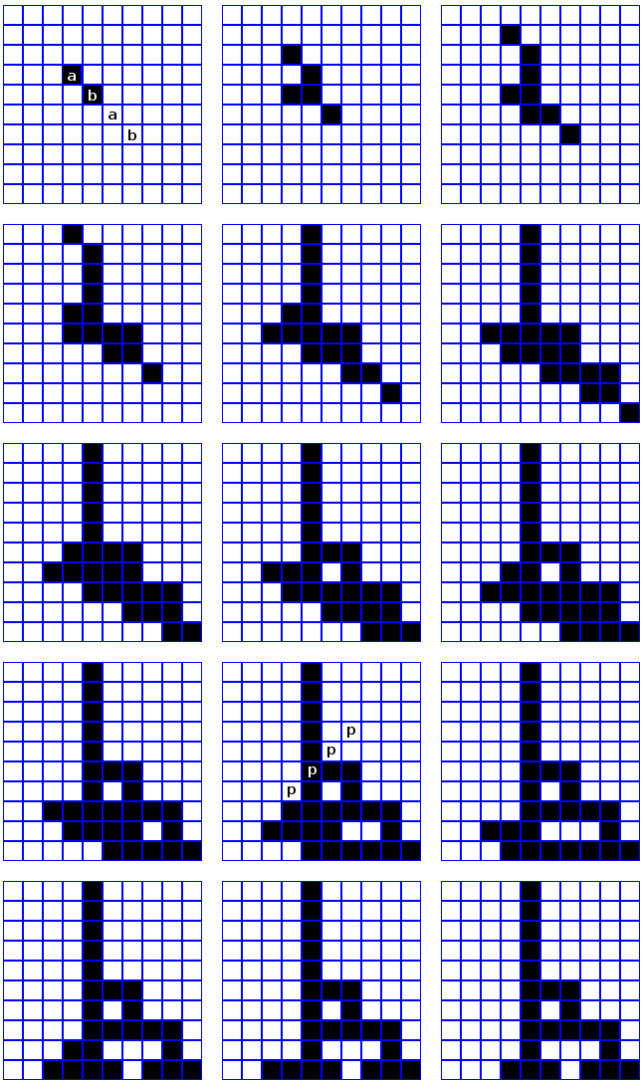


Fig. 8. Example of the development of CA performing the multiplication $2 \times 2 = 4$ using a diagonal input and output cell arrangement. White cell in the CA represents the state 0, black cell represents the state 1. The input operands are considered in the direct binary representation using the cell states ($a = b = (10)_{bin} = (2)_{dec}$, the product $p = (0100)_{bin} = (4)_{dec}$). The operand bits are interpreted at the main diagonal from MSB to LSB (from top-left to bottom-right), the product is interpreted at the antidiagonal from MSB to LSB (from bottom-left to top-right).

*	*	*	<=0	==0	*	>=1	*	<=0	0
==0	!=1	*	*	*	*	<=1	==1	!=1	1
!=1	==0	==0	<=1	*	*	>=0	<=1	*	0
==1	>=1	!=0	==1	*	<=1	>=1	*	>=1	0
!=0	<=0	*	*	>=0	<=0	<=0	==0	<=1	1
<=1	!=1	<=0	*	<=0	>=0	!=1	>=0	==1	1
!=1	<=0	==1	*	==1	!=0	==1	*	*	0
>=0	>=0	*	!=1	*	!=0	*	*	==0	1

Fig. 9. Example of a CMR-based transition function discovered by GA using the function set from Table I. This transition function realizes multiplication in the CA from Figure 8. One row in this figure represents a CMR. The converted table-based representation consists of 73 conventional rules.

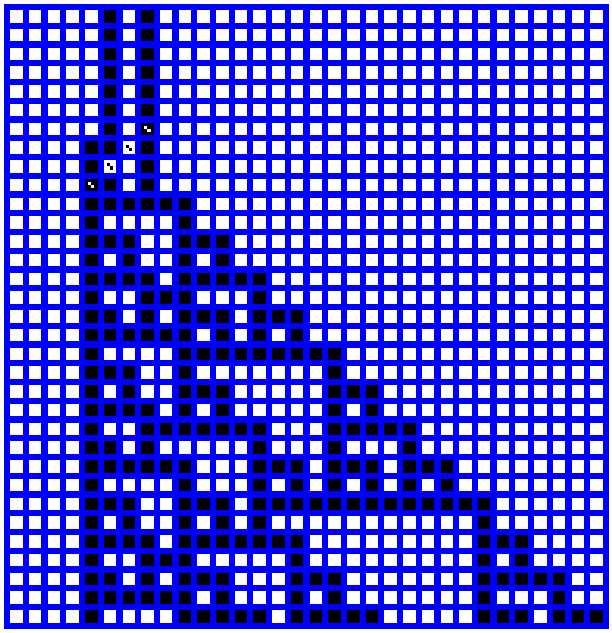


Fig. 10. Emergent behavior of the CA controlled by the transition function from Figure 9. The result of the product $3 \times 3 = 9$ is shown within the cells marked by two dots in the upper-left part of the CA. The diagonal cell setup was considered according to Figure 3c.

using conditionally matching rules. We focused on a case study whose objective was to design CA that are able to perform multiplication of two 2-bit operands. Several sets of experiments were presented considering various setups of both the evolutionary system (using the genetic algorithm) and cellular automata. In particular, two different sets of condition functions for realizing the transition rules of the CA were investigated in combination with various input and output cell arrangements in the CA. The experiments showed that the success rate of discovering a working solution is highly dependent on that cell arrangement. Moreover it also depends on the set of functions considered for the conditionally matching rules. Surprisingly, the experiments exhibit lower success rate in case of the condition functions that were identified in a separate experiment as potentially promising to solve a wide range of tasks. However, all sets of experiments provided some working solutions with interesting features.

The first is that the evolved transition functions are not limited to a CA of a specific size (i.e. the considered multiplication task may perfectly be solved in a larger CA even with more areas with the input and output cell structures). The second interesting phenomenon is the emergent global behavior (developed patterns) of the resulting CA in which a well-organized structures can be observed in addition to the primary computational task. A detailed analysis of some evolved results showed that such well-organized patterns of various complexity is a common feature of CA exhibiting the required computation. An interesting fact of these patterns is that they are very similar to some typical structures generated by one-dimensional cellular automata.

One of the questions that may arise from this investigation may be whether the process of multiplication could be simplified or optimized (e.g. to minimize the number of rules of the converted conventional transition function). Since massive parallel and homogeneous platforms may be very important in the future (especially in nanotechnology, molecular computing systems and similar areas), the principles of elementary approaches to studying the design, optimization and functioning of such systems are definitely worth the subsequent research.

The observations from the obtained results propose some issues for the future research in this area. For example, the selection of optimal set of condition functions in order to effectively design CA for solving a specific task still represents a challenging task. The in-depth analysis of the evolved functions and the possibilities of their applications in a wider context of the cellular platforms will also be investigated. Our ongoing experiments are devoted to the research of conditionally matching rules for efficient design of non-binary cellular automata.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation project 14-04197S, BUT project FIT-S-14-2297 and the IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070, funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, as well as Czech Ministry of Education, Youth and Sports via the project Large Research, Development and Innovations Infrastructures (LM2011033).

REFERENCES

[1] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.
[2] M. Bidlo and Z. Vasicek, "Evolution of cellular automata with conditionally matching rules," in *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*. IEEE Computer Society, 2013, pp. 1178–1185.
[3] S. Wolfram, *A New Kind of Science*. Champaign IL: Wolfram Media, 2002.
[4] M. Sipper, *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194*. Berlin: Springer-Verlag, 1997.
[5] G. Tufte and P. C. Haddow, "Towards development on a silicon-based cellular computing machine," *Natural Computing*, vol. 4, no. 4, pp. 387–416, 2005.
[6] S. Kumar and P. J. Bentley (eds.), *On Growth, Form and Computers*. Elsevier Academic Press, 2003.
[7] E. F. Codd, *Cellular Automata*. Academic Press, New York, 1968.
[8] K. Lindgren and M. G. Nordahl, "Universal computation in simple one-dimensional cellular automata," *Complex Systems*, vol. 4, no. 3, pp. 299–318, 1990.
[9] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays, Vol. 2*. A. K. Peters/CRC Press, 1982.
[10] C. G. Langton, "Studying artificial life with cellular automata," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 120–149, 1986.
[11] A. Gajardo, A. Moreira, and E. Goles, "Complexity of langton's ant," *Discrete Applied Mathematics*, vol. 117, no. 1–3, pp. 41–50, 2002.
[12] C. G. Langton, "Self-reproduction in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1–2, pp. 135–144, 1984.
[13] G. Tempesti, "A new self-reproducing cellular automaton capable of construction and computation," in *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life*, ser. Lecture Notes in Artificial Intelligence, vol. 929. Springer Verlag, 1995, pp. 555–563.

Appendix IV

On Routine Evolution of Complex Cellular Automata

BIDLO Michal

In: *IEEE Transactions on Evolutionary Computation, vol. 20, no. 5*, 2016, pp. 742-754. ISSN 1089-778X.

Journal Impact Factor in the year of publication: 10.629; This work was **awarded by Silver Medal** in 2016 Annual "Humies" Awards For Human-Competitive Results[1], a prestigious international competition held as a part of the Genetic and Evolutionary Computation Conference (GECCO), Denver, CO, US, 2016.

[1] http://gecco-2016.sigevo.org/index.html/Humies.html
https://www.human-competitive.org/awards

Appendix IV

On Routine Evolution of Complex Cellular Automata

Michal Bidlo

***Abstract*—This paper discusses a special technique, called conditionally matching rules (CMRs), for the representation of transition functions of cellular automata (CA) and its application to the evolutionary design of complex multistate CA. The problem of designing replicating loops in 2-D CA and the square calculation in 1-D CA will be treated as case studies. It will be shown that the evolutionary algorithm in combination with CMRs is able to successfully solve these tasks and provide some innovative results compared to existing solutions. In particular, a novel replication scheme will be presented that exhibits a higher replication speed in comparison with the existing replicating loops. As regards the square calculation, some results have been obtained that allow a substantial reduction of the number of steps of the cellular automaton against the currently known solution. The utilization of the CMRs in the proposed experiments represents the first case of a successful automatic evolutionary design of complex CA for solving nontrivial problems in which the existing conventional approaches have failed.**

***Index Terms*—Cellular automaton, evolutionary algorithm (EA), replicating loop, square calculation.**

I. Introduction

SINCE the introduction of cellular automata (CA) in [1], researchers have dealt with the problem of how to effectively design a cellular automaton (and its transition function in particular) to solve a given task. Although many successful applications of CA have so far been presented in various domains (e.g., concerning artificial life [2], [3], nano-computing [4], image processing [5], [6], molecular simulations [7], or even the utilization of DNA (deoxyribonucleic acid) molecules to constructing nano-scale CA-based computing devices [8] including some applications of this concept [9]), the process of designing suitable rules to solve specific problems in CA still represents a difficult task.

This paper deals with the evolutionary design of CA using a special representation technique referred to as conditionally matching rules (CMRs). The basic structure of a cellular automaton assumes a regular structure of cells, each of which at a given moment occurs in a state from a finite set of states. The behavior (or development) of a CA will be considered as a synchronous update of the cell states according to a transition function in discrete time steps. It will also been assumed that the states are discrete (integer) values. The transition function determines the next state of a cell depending on the combination of states in a cellular neighborhood that includes the cell to be updated and its neighbors. There are two fundamental concepts of CA as regards its transition function.

1) The basic (uniform) CA work with cells that share a single transition function. In this case, the transition function of a cell can be considered as the transition function of the CA.
2) The nonuniform concept allows individual cells to determine their states according to different (local) transition functions.

In both cases the behavior of the CA arises from a cooperative update of all its cells during a sequence of development steps. The design of a suitable (efficient) transition function represents a key process aimed at achieving the desired behavior of a CA. The cellular neighborhood is defined uniformly for each cell and its form primarily depends on the dimension of the CA. In this paper, 1-D and 2-D uniform CA will be considered whose behavior is controlled by a deterministic transition function and the cellular neighborhood is defined as follows. In the case of the 1-D CA the neighborhood of each cell is composed of the given cell and its immediate left and right neighbor (a three-cell neighborhood). Regarded as the cellular neighborhood of the 2-D CA will be the given cell and its immediate neighboring cells in the north, south, east, and west directions (a five-cell neighborhood). Cyclic boundary conditions will be applied due to the limitation of the CA size to a finite number of cells only for practical implementations. This means that the left-most and right-most cells in the 1-D CA are considered as neighbors and similarly, in the case of the 2-D CA, the opposite cells at the boundary of the cellular array in both dimensions are considered as neighbors. The CA will be assumed to work with more than two cell states and referred to as multistate CA.

A. Overview of Cellular Automata Literature

Many CA-based systems were successfully designed using analytical methods (for example, for the investigation into computational properties of the CA and construction of computing systems [10]–[15], development of replicating structures [16]–[19], or solving some specific mathematical operations and benchmarks in the cellular space [20]–[24]).

Manuscript received June 4, 2015; revised September 17, 2015 and December 20, 2015; accepted December 24, 2015. Date of publication January 8, 2016; date of current version September 30, 2016. This work was supported in part by the Czech Science Foundation through the Advanced Methods for Evolutionary Design of Complex Digital Circuits project GA14-04197S.

The author is with the IT4Innovations Centre of Excellence, Faculty of Information Technology, Brno University of Technology, Brno 61266, Czech Republic (e-mail: bidlom@fit.vutbr.cz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2016.2516242

Appendix IV

However, the process of determining a suitable transition function for a given application represents a difficult task, especially due to an enormous growth of the solution space in dependence on the number of cell states, and due to the fact that the process of "programming" the CA is not intuitive. In order to overcome this issue, the aim is to automate the process of designing (or identifying) the transition rules, using both deterministic algorithms and other heuristics or unconventional (nondeterministic) techniques, including evolutionary algorithms (EAs).

Adamatzky [25] proposed an algorithm for the identification of CA rules from a given series of global transformations during a finite number of CA steps. His approach was later improved, e.g., by combining it with learning classifier systems [26] or advanced representation techniques like polynomial representation or decision trees [27]. Holland [28], Packard [29], and Richards *et al*. [30] were among the first who applied genetic algorithm (GA) in order to adapt the transition rules. Sapin *et al.* [31] used an evolutionary approach to study the problem of discovering gliders in a cellular array. Sipper [32] proposed a special technique, called cellular programming, for a parallel evolution of nonuniform CA, using a modified GA in each cell. In recent years, several works were published dealing with the design of CA using, various evolutionary techniques. For example, Breukelaar and Bäck [33] applied GA in order to evolve multidimensional uniform CA to solve the density task and checker-board benchmarks. They used ordinary table-based representation of the transition function and focused on tuning the GA settings, concluding that their system performed better than that in an earlier work published by Mitchell *et al.* [34]. Sapin [35] investigated the evolutionary discovery of glider guns in CA. Elmenreich and Fehérvári [37] proposed an original technique for calculating the transition function of CA, using neural networks (NNs). The goal was to train the NN by means of evolutionary programming [36] in order to develop self-organizing structures in the CA [37]. In addition, various advanced concepts and modifications of CA were investigated. For instance, Medernach *et al.* [38] studied a heterogeneous concept of CA whose cells utilize some advanced items like age, decay, or genetic transfer using open-ended evolution to create an evolving ecosystem of competing cell colonies. Bandini *et al.* [39] dealt with effects in CA that may be observed by introducing asynchronous update schemes.

As regards research into multistate CA in recent years, several studies using various bio-inspired techniques have been proposed. For example, in [40] a swarm intelligence algorithm, called stochastic diffusion search, was applied as a tool to identify symmetry axes in patterns generated by CA. The results provided a deeper insight into the emergent behavior of CA and showed some interesting features (e.g., aesthetic qualities) of the generated patterns with potential applications in computer graphics. Javid, al-Rifaie, and Zimmer [40] say "a 2-D multistate cellular automaton with periodic boundary provides an endless environment for the growth of patterns and the observation of emergent complex behavior over the time of evolution." This statement indicates the importance of (multistate) CA research since the results may provide valuable information for the area of complex systems in general (especially the issue of emergent behavior, which allows using simple rules in order to achieve a complex, cooperative global behavior). Skaruz *et al.* [41] published an approach to pattern and image reconstruction, using multistate CA. They applied the three-state 2-D CA with nine-cell neighborhood and showed that the GA was able to discover satisfactory transition rules to reconstruct an image with up to 70% of damaged pixels. It is worth noting, however, that a certain margin of error can be tolerated in the case of image processing, which may reduce the complexity of searching for the rules. Baetens and De Baets [42] studied the issue of stability and defect propagation in the development of 1-D three-state CA with three-cell neighborhood. In particular, it was shown how the assessment of stability could be performed using nondirectional Lyapunov exponents (based on a previous study proposed in [43]). The authors focused on a special class, called totalistic CA, in which the rules depend only on the total (average) of the cell states in a neighborhood.

The main focus of this paper (in comparison with the aforementioned studies) is on investigating both 1-D and 2-D CA working with at least six cell states. The transition function (to be discovered by EA) is not limited to any specific class of CA (i.e., any arbitrary solution is acceptable that satisfies the conditions, i.e., the behavior specified for the CA). The CA behavior (evaluated in the fitness function of EA) will be exactly given (for example, as a minimal number of copies of a specific structure required to emerge within a given finite number of CA steps, an exact pattern to be developed from a given initial CA state, the result of a calculation encoded in a stable final CA state with respect to the initial state).

B. Motivation and Goals

Cellular systems demonstrated some advantageous features in solving various problems of both application-specific and generic nature. Although some automatic design techniques for CA have been proposed and successfully validated on selected case studies and benchmarks, some limitations can be observed if a solution of a different kind or more general problem is needed. For example, cellular programming cannot effectively handle uniform CA that may be more interesting from the viewpoint of the physical implementation or control than the nonuniform model. The identification of the transition rules from a CA development sequence is not applicable if the behavior of the CA is not known exactly. For the increasing number of cell states the solution space of the CA grows significantly, which makes the search for a suitable transition function difficult (the problem of scale). But it is not only for these reasons that the research into new (unconventional) design methods and representations is still important.

Considering the recent progress in physical theory and information technology, advanced models have been studied involving principles from quantum theory [44], [45], nanoscale design [46], [47], implementation on the molecular level [48], or combination of some of these principles for the field-programmable gate array design [49]. Another example could be the evolution of transport networks using CA models

Appendix IV

inspired by slime mold of a large cell called Physarum polycephalum [50]. This shows that CA have become a model applicable in current interdisciplinary areas for which new findings will be important even from the research on the elementary level.

The following scenarios can be identified regarding the design of transition rules for CA.

1) *Identification of the Transition Rules From a CA Analysis:* This method assumes that a sequence of CA states is known (i.e., the CA development for a finite number of steps). The task is to identify the transition rules according to which the cells update their states.
2) *Design of the Transition Rules for Given Conditions in CA:* The exact CA behavior is not known, the task is to design a transition function together with (a part of) the CA development that fulfils the given requirements (e.g., to achieve a specific state from a known initial state, to achieve a periodic or stable behavior, etc.). This scenario will be considered in this paper.

The goal of this paper is to investigate the automatic evolutionary design of complex multistate CA, using an EA combined with a special technique to encode the transition functions referred to as CMRs. In order to demonstrate the abilities of this method, several different conditions required and evaluated in the CA will be considered. In particular, the problem of generic square calculation in 1-D CA will be presented as the first case study, whose results show that it is possible to substantially reduce the number of steps needed to calculate the square in comparison with the existing solution. The second (more complex) study will investigate the evolution of nontrivial replication processes in 2-D CA taking into consideration three different loop-like structures. It will be shown that more efficient results can be generated using the CMRs in comparison with the existing solutions. A novel replication scheme will be presented that exhibits a higher replication speed in comparison with the existing replicating loops. The utilization of the CMRs in this paper represents the first case of a successful evolutionary search for solutions to problems in multistate CA for which the conventional approaches have failed.

II. CA Evolution Using Conditional Rules

For a successful CA design the representation (encoding) of the transition function represents a key issue. This section summarizes the most important representations known from the literature and describes the principles and setup of CMRs—an advanced representation proposed for the evolutionary design of multistate CA—that will be considered for the experiments in this paper.

A. Overview of Representations for Cellular Automata

A common (basic) approach to representing the transition rules is a table-based method where a rule specifies a new state for a given (single) combination of states in the cellular neighborhood. For example, Packard [29], Richards *et al.* [30], Sipper [32], or Breukelaar and Bäck [33] used this kind of encoding for the evolutionary design of CA. However, the table-based method is not suitable for multistate CA because the complexity of designing such CA increases significantly with increasing number of states. Therefore, advanced representations have been investigated. Andre *et al.* [52], [53] utilized genetic programming (GP) [52], in which the transition function is encoded in a tree representing a program that calculates the updated cell states. Other specific representations have been used for the identification of CA rules from a sequence of global states, e.g., polynomial representations [54] or decision trees [55]. The advanced representation techniques can provide some improvements over the basic approach, e.g., a reduction of the size of representation and computational complexity or increasing the modification (manipulation) efficiency. In particular, Bidlo and Vasicek [57] used a representation based on linear GP [56] to evolve CA and demonstrated an increased success rate and reduced computational effort over the table-based method for the replication and pattern development problem. However, advanced experiments showed that this method was not suitable for more complex problems in which specific transition rules need to be applied. Therefore, the concept of CMRs was introduced as described in the following section.

B. Conditionally Matching Rules

CMRs were introduced in [58] and their abilities demonstrated when solving a specific kind of the replication problem and a nontrivial pattern transformation problem in binary 2-D CA, where the GP-based representation failed. Moreover, the potential of this method was demonstrated by further successful experiments regarding the evolution of multiplication in the uniform 2-D cellular array [59]. The following paragraphs describe the setup of the CMRs that will be applied to the evolutionary design of multistate CA.

A conditionally matching rule (CMR) represents a generalized rule of a transition function for determining a new cell state (in view of the table rules). Whilst, the basic transition rule specifies a new state for a specific combination of states in the cellular neighborhood, a single CMR may cover more than one combination. A CMR is composed of two parts: 1) a conditional part and 2) a new state. The number of items (size) of the conditional part corresponds to the number of cells in the cellular neighborhood. Let us define the condition item as an ordered pair of a condition and a state value. The condition is typically expressed as a function whose result can be interpreted either as true or false. The condition function evaluates the state value in the condition item with respect to the state of the appropriate cell in the cellular neighborhood. In particular, each item of the conditional part is associated with a cell in the neighborhood with respect to which the condition is evaluated. If the result of the evaluation is true, then the condition item is said to match with the cell state in the neighborhood. In order to determine a new cell state according to a given CMR, all its condition items must match (in such a case the CMR is said to match). Fig. 1 shows an example of a CMR defined for a 2-D CA with five-cell neighborhood where ordinary relational operators $==$, $\neq$, $\leq$, and $\geq$ are considered as the condition functions. Note that these operators will be considered for all the experiments presented in this paper.

Appendix IV

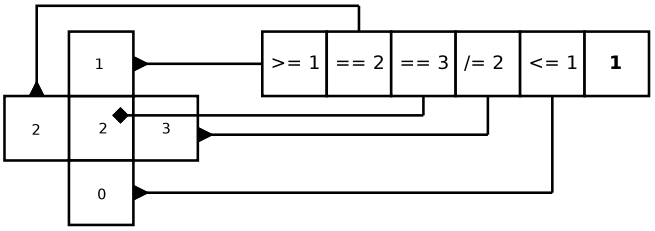


Fig. 1. Example of a conditionally matching rule working with four cell states and the five-cell neighborhood. The right-most value of the CMR represents the new state (shown in bold).

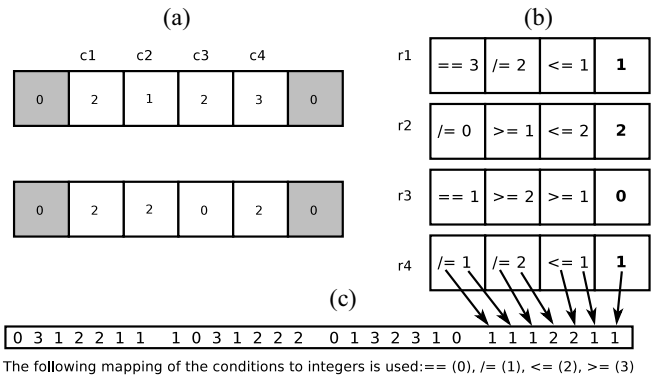


Fig. 2. (a) Example of a multistate 1-D CA working with four cell states, a three-cell neighborhood, and zero boundary conditions. Single step performed according to a (b) CMR-based transition function is illustrated. (c) Corresponding chromosome of this transition function together with encoding the condition functions as integer values.

A CMR-based transition function is considered as a finite sequence of CMRs. The algorithm of determining a new state is the following. The CMRs are evaluated sequentially, starting with the first CMR in the sequence. If a CMR compares the given cellular neighborhood, then this CMR is used to determine the new state of the cell to be updated. Note that, because of the sequential evaluation of the CMRs, it is always the first matching CMR in the sequence. If none of the CMRs matches, then the cell keeps its current state. This approach ensures that the process of calculating the new state is deterministic (on the assumption that the condition functions are deterministic too). The aspect of determinism is important in order to preserve the traditional concept of CA, and with respect to the applications investigated in this paper. In the case of the deterministic CA, the process of development ensures that for a given configuration the CA produces a specific behavior (result of the development). Specifically, the result of the square calculation has to be the same for a given value, and for the replication of a loop it is expected the copy is the same as the original structure. Moreover, a deterministic (CMR-based) transition function can be transformed into a corresponding table-based representation, which allows us to use a conventional form of the transition rules implemented in many existing CA simulators. Another advantageous feature of the CMRs is the ability to specify conventional (table-based) rules if needed. In order to do that, the relation == with given state values specified in each item of the conditional part of a CMR allows specifying a transition rule for a given (single) combination of states in the cellular neighborhood.

An example of a 1-D CA step according to a sample CMR-based transition function is shown in Fig. 2(a) and (b). Note that the cell states are updated synchronously, i.e., the cells evaluate the transition function in parallel with respect to the actual CA state (2 1 2 3). Cell $c1$ will not change its state because no CMR matches the cellular neighborhood 021. For cell $c2$ only CMR $r2$ matches the neighborhood 212, therefore, the new state of this cell will be 2. Cell $c3$ will get its new state according to CMR $r3$, which matches the neighborhood 123. Finally, rules $r2$ and $r4$ match the neighborhood 230 of cell $c4$, therefore CMR $r2$ will change the state of $c4$ to 2.

Considering the aforementioned example with a four-state CA and three-cell neighborhood, the complete table of the transition function consists of $4^3 = 64$ rules (each rule for a single specific combination of states in the cellular neighborhood). This means there are in total 4^{64} possible transition functions (possibilities of what next state can be specified for each combination). However, the CA often involves only a subset of rules—combinations of neighborhood states—for which a different state is specified with respect to the current state of the cell to be updated (these rules need to be explicitly specified in the transition function). For the purposes of the evolutionary design it would be possible to shorten the chromosomes of the EA (and thus to reduce the search space) if the subset of explicit rules were known. Unfortunately, it is difficult to effectively predict these rules for a given application if the CA development is not known exactly. In order to reduce the size of the chromosomes needed to encode the transition function, the CMR approach was introduced. If the example from Fig. 2 is considered (four cell states, three-cell neighborhood, and four condition functions), each CMR can be encoded as a 7-tuple of integers. For a transition function consisting of four CMRs, a candidate solution can be represented by $4 \times 7 = 28$ integers, each of which may acquire four different values. Therefore, the search space can be reduced substantially to 4^{28} possible solutions. Our hypothesis is that the CMRs allow exploring more effectively the search space and discovering solutions to some problems that were not achieved using the conventional representation.

C. Design of CMR-Based CA Using Evolutionary Algorithm

In this paper, the search for a suitable sequence of CMRs satisfying a given CA behavior is performed using a population-based EA. The EA has been chosen after a long-term experience with the CA design, and its setup is based on that proposed in [58]. Our preliminary experiments with this EA showed that its simple concept and computationally efficient mutation operator represent the biggest advantages (potentially suitable for accelerated hardware implementations in the future). The small population size allows performing a satisfactory amount of iterations (hundreds of thousands to several millions, depending on the specific experiment) in order to evolve the candidate solutions whose evaluation takes a long time with respect to the computational effort of the EA (this holds for the evaluation of CA performing several tens of development steps). Note that a detailed investigation and

Appendix IV

optimization of the EA operation are not the subject of this paper.

A population of eight chromosomes is considered, each of which represents a candidate transition function encoded as a finite sequence of CMRs. Each CMR is encoded as a finite sequence of integers representing the conditional parts (i.e., the condition functions and state values) and the next state as illustrated in Fig. 2(c). Note that the chromosome is represented as a linear array of integers directly coding the condition functions and state values for each CMR. The chromosomes for the initial population are generated randomly at the beginning of the evolution. The evaluation of each chromosome is performed using a CA whose development (controlled by the transition function encoded in the chromosome) from a given initial state is observed with respect to the required behavior for a finite number of steps. The objective function that calculates the fitness values of the chromosomes is specific for different case studies and will be described for each experiment in Sections III and IV.

In each generation of the EA a new population is created according to the following algorithm: four chromosomes are selected randomly, the best one of which is considered as a parent (i.e., fitter chromosomes are preferred to generate offspring)—it is a case of the tournament selection with base 4. An offspring is created by mutating 0–2 randomly selected integers in the parent, which is performed via replacing the selected integers by new valid randomly generated values. The number of integers to be mutated is also selected randomly; if 0 integers are chosen, then the offspring is identical to the parent. The chromosome selection and mutation continue until the new population of the same size is filled by the offspring. If a solution is found (after evaluating the chromosomes in the new population) that satisfies the given CA behavior, then the evolution is considered as successful. If no solution is found within a maximal number of generations (which is specific to various experiments), then the evolution is terminated.

III. Experiments With Square Calculations

The first case study considers the evolutionary design of 1-D CA whose development can be interpreted as calculating the square of a number. The basic idea to perform this operation in the CA (together with choosing the appropriate number of cell states, representation of input values, and way of evaluating candidate solutions) has been inspired by Wolfram's work [23, Section Computations in Cellular Automata, p. 638]. The computation of x^2 is interpreted as a CA development that comes (after a finite number of steps) into a stable state in which the result for the given x is encoded.

In the proposed experiments a 1-D CA consisting of 100 cells and working with eight cell states will be considered. The value of x will be encoded in the initial CA state as a continuous sequence of cells in state 1, whose length corresponds to x, the other cells possess state 0. For example, a ten-cell CA encoding $x = 3$ can appear as 0001110000. The result is assumed to be a stable CA state in which a continuous sequence of cells of a single state different from state 0 can be detected whose number equals x^2, the other cells are required to be in state 0. The goal is to discover CA that are able to calculate the square of an arbitrary number $x > 1$.

TABLE I
Statistics of the Evolutionary Experiments Dealing With the Calculation of x^2 in 1-D CA. This Set of Experiments Considered CA Working With Eight Cell States

x eval. →	from 2 to 5			from 2 to 6		
#CMRs	Succ. rate	Avg. #gen.	Std. dev.	Suc. rate	Avg. #gen.	Std. dev.
20	57	60709	54704	26	72648	48215
30	64	60293	50602	28	79299	65138
40	59	67539	55114	26	85865	57281
50	57	61342	52561	32	80526	51720
#general		4			6	

Two scenarios of the fitness evaluation are investigated: in the first case the values of x from 2 to 5 are evaluated during the evolution whilst the second setup considers x from 2 to 6. This approach is motivated by the assumption that if the CA is required to work for more values of x during the evolution, then it will be harder for the EA to design such a CA (i.e., the success rate will be lower) but, on the other hand, more general solutions could be obtained (i.e., such CA that, using the evolved transition function, are able to correctly calculate the square for any higher number not considered in the fitness evaluation). The result of the x^2 calculation in the CA is evaluated after the 99th and 100th steps in order to determine whether the resulting state is stable. Considering the aforementioned setup the fitness of a fully working solution for x from 2 to 5 is given by $F_{\max 2-5} = 4{*}2{*}100 = 800$ (4 different values of x are considered, the result of each is evaluated for the last two steps in a CA consisting of 100 cells), the second scenario considers the maximal fitness $F_{\max 2-6} = 5 * 2 * 100 = 1000$. The maximal limit of generations for these experiments was set to 200 000. For each scenario 100 independent evolutionary runs were executed. The success rate and the average number of generations needed to find a working solution were measured. In order to identify general solutions for the square calculation, the resulting CA were validated for the values of x up to 100, using the transition functions obtained from the successful evolutionary runs. For the purposes of this paper the solution that passed this test is considered as general.

Table I summarizes the statistics of the evolution for this set of experiments. As evident, the success rate is substantially lower if more values of x (i.e., from 2 to 6) are evaluated. In this case, approximately half the evolutionary runs finished successfully in comparison with the scenario with x from 2 to 5. However, more general solutions were obtained as shown in the last row of Table I, which confirms the assumption stated in the previous paragraph. Although the general square calculation definitely cannot be regarded as a trivial task for uniform CA, the success rate and the resulting number of generations indicate that the proposed CMR encoding of the transition functions represents an efficient technique of solving this problem for reasonable values of x considered during the fitness evaluation.

Appendix IV

TABLE II
ANALYSIS OF SELECTED RESULTS FOR THE SQUARE CALCULATION. THE NUMBER OF STEPS OF THE CA NEEDED TO OBTAIN THE RESULT OF x^2 FOR x FROM 2 TO 9 USING THE BEST OBTAINED RESULTS AND WOLFRAM'S SOLUTION [23] ARE SHOWN

x	2	3	4	5	6	7	8	9
Solution #1, 400 transition rules								
#steps	6	12	24	40	60	84	112	144
Solution #2, 408 transition rules								
#steps	5	12	20	34	48	68	88	114
Solution #3, 432 transition rules								
#steps	4	10	20	32	48	66	88	112
Wolfram's solution								
#steps	12	28	50	78	– not available –			

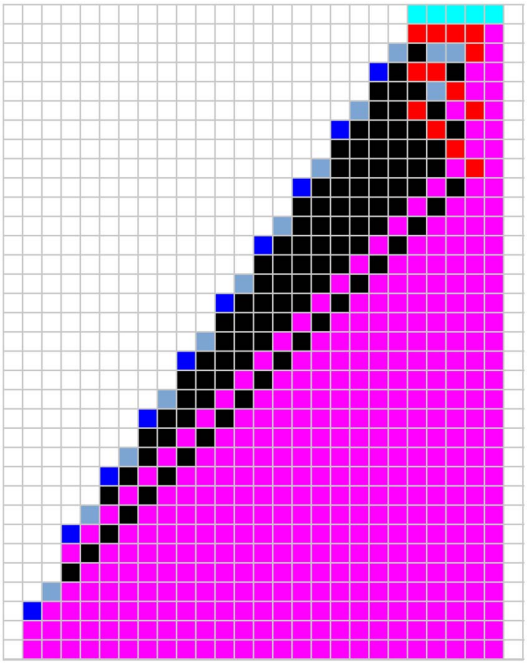


Fig. 3. Calculation of x^2 in one of the best CA discovered by evolution ($x = 5$ in this example). This CA works with 432 transition rules, 32 steps are needed to obtain the result (note that Wolfram's solution [23] needs 78 steps).

Table II shows the number of steps of the best evolved solutions needed to obtain the result of x^2 in the CA. The results obtained are compared to Wolfram's solution published in [23]. The number of CA steps determines the efficiency of the square calculation (i.e., the fewer steps, the faster the calculation). From this point of view the best result obtained in this paper (denoted as Solution #3 in Table II, whose CA is controlled by 432 rules) is more than twice faster in comparison with Wolfram's CA (see the last row of Table II). For example, Solution #3 calculates 5^2 in 32 steps whilst Wolfram's CA needs 78 steps. An example calculating 5^2 using our best solution is illustrated in Fig. 3.

An analysis of the results has shown that all the general solutions exhibit a regular pattern generated by the CA during its development (which is actually inevitable in order to correctly calculate the result for various x using the same transition function of the CA). However, it can be observed that various approaches can be discovered that differ in both the efficiency and the complexity of the CA. Moreover, one of the key features is the way of encoding the input value and the result. Whilst Wolfram used a method of representing these values by two different (nonzero) states in the CA, the encoding proposed in this paper considers only a single state and allows the result to be represented by a different state than the input value is encoded by (which probably enabled a more efficient square calculation in comparison with Wolfram's solution). These observations indicate that research into advanced techniques of input and output representation could provide further interesting solutions to this problem.

IV. EXPERIMENTS WITH REPLICATING LOOPS

The second case study considers a class of 2-D CA that are able to replicate a given structure. Replicating loops represent one of the typical benchmarks that have been studied in relation with CA. There are various variants of loops that differ in shape, size, complexity, or replication speed. In most cases CA that perform replication of the currently known loops were designed using analytical approaches (i.e., the transition rules were specified manually after an in-depth understanding of the loop structure and the transformation process leading to the creation of its copy). In this section, we demonstrate that it is possible to automatically design transition functions for various replicating loops. In particular, it will be shown that the EA can discover a completely new replication scenario whose replication speed is several times higher in comparison with some currently known loops. Note that for the purposes of this paper the replication speed will be considered as the number of copies of the loop that can be created depending on the number of CA steps executed. The reason for the utilization of this metric is that it allows quantifying an amount of objects (e.g., loop structures) the CA is able to create in a given number of steps and comparing its behavior with other known replicating structures. In general, the speed of the CA development could, for example, express the area of the cellular array the CA is able to cover (or to process), taking into consideration some constraints and conditions of a specific application.

Experiments with the replicating loops were conducted using the EA described in Section II-C, for which the maximal number of generations was set to 3 million. The evolution time of a single run is approximately 12 h when using the Anselm cluster, which is a part of the Czech National Supercomputing Center.[1]

Several sets of experiments were conducted with various replicating loops. Fig. 4(a)–(c) shows the loops that were considered in our experiments. For the purposes of this paper the loops will be denominated by symbolic names as round loop [Fig. 4(a)], rectangular loop [Fig. 4(b)], and envelope loop [Fig. 4(c)]. Note that the loop shapes as well as the numbers of cell states proposed for specific experiments were chosen on the basis of the existing replicating loops. In addition, the

[1] http://www.it4i.cz/?lang=en
https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

Appendix IV

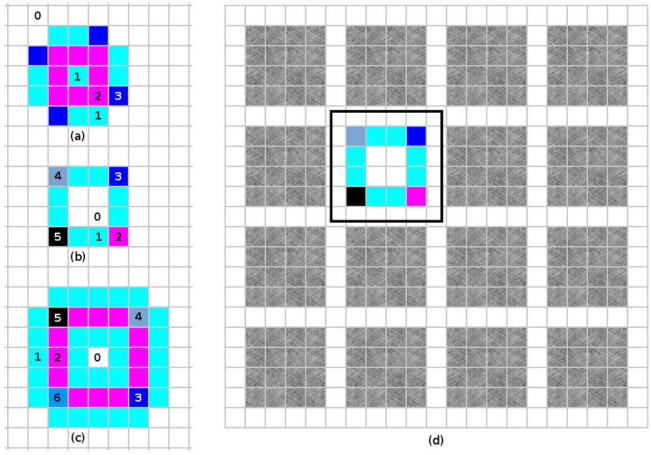


Fig. 4. Loop structures considered for the replication experiments (the corresponding state values of cells are also shown). (a) Round loop. (b) Rectangular loop. (c) Envelope loop. (d) Illustration of a fitness calculation scheme used for the round and the rectangular loop. The textured cells are evaluated with respect to the required replica structure. A boundary of white cells in state 0 separates the replicas. The initial loop is marked by a thick rectangle.

objective is to determine the ability of the EA and CMRs to tackle various ways of the fitness evaluation and specification of the target CA behavior.

The round loop and rectangular loop share a common principle of the fitness evaluation based on the known shape and size of the loop. Fig. 4(d) illustrates the concept of the fitness evaluation containing a rectangular loop as an example (note that the CA size was chosen with respect to a reasonable space for creating the replicas in various directions). As shown by the textured cells, the replicas are expected to be in a regular grid with respect to the initial loop. This scenario was applied on the basis of some existing replicating loops that work in a similar way. The replicas are required to be separated from each other by a boundary consisting of cells in state 0. Thus, in the case of the rectangular loop in Fig. 4(d) the complete rectangle of a replica to be evaluated consists of 6×6 cells as marked by a thick rectangle. In order to evaluate a candidate solution, the CA development is analyzed for 30 steps. After each step of the CA the following calculation is performed. The partial fitness is calculated separately for each replica in the grid as the number of cells in correct state with respect to the loop rectangle. Then the step fitness is defined as a sum of all partial fitness values for a given step of the CA as follows. If the partial fitness equals the number of cells in a given rectangle (i.e., a perfect replica is detected), then the step fitness is increased by the double of this number (as a bonus for finding a replica), otherwise the step fitness is increased by the original partial fitness value. The final fitness is determined as the maximal step fitness out of all the CA steps considered for the evaluation. If at least four perfect replicas are detected in a state within the 30-step development, the evolution is finished successfully.

In order to validate the results, a software simulator developed by the author of this paper is applied. A larger CA is executed for more than 30 steps using the evolved transition functions, and a visual inspection is performed in order to identify general replicators. For the purposes of this paper, each solution with the ability to persistently produce replicas according to the specification in the fitness function is classified as a general replicator.

Statistical results of the replication experiments are summarized in Table III. For each type of the replicating loop the success rate and computational effort (expressed as the average number of generations needed to find a working solution) were evaluated depending on various selected CMR counts. The number of general solutions is determined out of all successful experiments performed for a given loop. Setups with six and eight cell states were considered for the rectangular and the round loop. The results show that 30 or 40 CMRs are in most cases adequate to find a working solution. For 20 CMRs the success rate is usually very low, probably due to a lack of resources (rules available for the CA) that can be expressed by the CMRs. On the other hand, 50 CMRs induced a huge search space that is very time consuming for the EA to explore (i.e., the success rate is low with a given generation limit). Although the success rate achieved in general is under 20%, these experiments showed for the first time the possibility of automatically designing complex multistate CA using the CMR encoding. Note that we were not able to successfully design any of the CA that use conventional representations of the transition functions.

The results in Table III also show that most of the experiments require on average more than a million generations in order to find a working solution. This means that the replication does not represent a trivial task for the EA so that a significant part of the search space needs to be explored. Although the population works with eight individuals only (this may justify the need for a higher number of generations), the time needed to evaluate the candidate solutions is considerable, which is not feasible for larger populations. The advantage of such a small population is also supported by the fact that in the proposed EA a single parent is chosen from four individuals (i.e., from half the population) and this parent produces offspring for the next generation. Experiments showed that a larger population did not improve the success rate because most of the time was spent on the fitness calculation, with no significant increase in the exploration of promising parts of the solution space. Several parameters can be tuned in the proposed EA: population size, tournament selection base, and the number of genes to be mutated. The optimal setting is usually problem dependent, which requires tuning the EA separately for different case studies. However, the tuning of the EA was not a goal of this paper—the objective was to evaluate the proposed EA with CMRs using several different tasks in CA. In the case of successful experiments the tuning of the EA and examination of its features for various problems can be performed, which represents a topic for the future research.

Most of the general results obtained in this paper replicate the given loop in a single direction only. However, in some cases the EA discovered a novel approach that can be used to effectively replicate the given structure from many (previously created) instances simultaneously. The best results obtained for each loop are described in the following sections.

Appendix IV

TABLE III
STATISTICS OF EXPERIMENTS DEALING WITH EVOLUTION OF THE REPLICATING STRUCTURES FROM FIG. 4. RESULTS ARE CALCULATED OUT OF 100 INDEPENDENT EVOLUTIONARY RUNS

Loop →	**Rectangular 6 states**			**Rectangular 8 states**			**Round 6 states**			**Round 8 states**			**Envelope 8 states**		
#CMRs	**Succ. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**
20	6	1320293	485774	7	1750754	714081	1	546936	-	3	1113809	983806	2	2655646	208081
30	5	1719899	871523	13	1404925	728908	12	1152757	674883	12	993487	631610	5	1626788	595484
40	12	1022548	815782	14	1056619	783897	12	804928	544980	19	867039	579131	11	1603042	706082
50	6	686951	410334	12	1195469	1110617	2	1402899	865896	10	841832	698793	9	976695	672486
#general	5			3			11			20			20		

N	W	C	E	S	new
<= 0;	!= 0;	!= 0;	!= 0;	== 0	6
<= 4;	>= 2;	== 0;	<= 6;	>= 6	6
== 0;	== 0;	!= 0;	== 0;	>= 7	5
<= 3;	>= 1;	<= 4;	<= 1;	>= 6	3
>= 0;	>= 4;	>= 0;	<= 0;	!= 0	1
>= 5;	>= 3;	!= 0;	<= 6;	== 0	0
<= 3;	<= 1;	== 0;	<= 1;	<= 1	0
>= 2;	!= 0;	!= 0;	<= 2;	== 0	1
== 0;	!= 0;	>= 1;	<= 3;	>= 3	7
== 0;	!= 0;	== 0;	!= 0;	<= 3	0
>= 6;	!= 0;	>= 7;	<= 7;	!= 0	3
<= 1;	== 0;	<= 6;	== 0;	<= 2	1
== 0;	>= 3;	!= 0;	<= 4;	<= 1	0
== 0;	!= 0;	== 0;	>= 7;	>= 1	2
<= 4;	>= 1;	== 0;	!= 0;	== 0	5
>= 3;	!= 0;	== 0;	<= 5;	>= 0	1
>= 1;	== 0;	>= 2;	>= 1;	<= 3	3
!= 0;	!= 0;	>= 2;	<= 3;	>= 1	2
>= 3;	<= 3;	!= 0;	!= 0;	!= 0	1
<= 5;	== 0;	<= 3;	== 0;	>= 7	2
<= 5;	<= 3;	== 0;	== 0;	== 0	2
<= 2;	!= 0;	>= 1;	>= 0;	<= 0	3
== 0;	<= 1;	>= 5;	== 0;	== 0	1
>= 1;	<= 3;	<= 6;	== 0;	== 0	7
>= 4;	<= 0;	>= 6;	<= 0;	<= 7	0
== 0;	>= 1;	>= 2;	>= 2;	<= 7	0
<= 1;	<= 6;	== 0;	<= 2;	== 0	1
!= 0;	!= 0;	<= 6;	!= 0;	== 0	7
>= 1;	!= 0;	!= 0;	!= 0;	<= 1	2
== 0;	== 0;	<= 3;	<= 2;	<= 1	3

(a)

N W C E S	new
0 0 0 0 2	1
0 0 0 0 7	2
0 0 0 2 0	1
0 0 0 2 1	3
0 0 1 0 7	5
0 0 1 2 1	3
0 0 2 0 0	1
0 0 2 2 0	3
0 0 5 0 2	1
0 0 7 0 0	1
0 1 0 0 6	3
0 1 0 1 6	3
0 1 1 0 3	7
0 1 1 1 3	7
0 1 1 1 7	3
0 1 2 0 0	3
0 1 3 1 3	7
0 1 3 2 0	6
0 1 3 2 3	7
0 1 6 0 0	3
0 1 6 1 6	7
0 1 7 5 3	0
0 1 7 7 1	0
0 2 0 0 0	2
0 2 0 0 6	6
0 2 0 3 7	6
0 2 2 2 1	0
0 2 6 0 7	7

N W C E S	new
0 2 6 3 7	7
0 3 0 0 0	2
0 3 1 0 1	0
0 3 1 0 5	7
0 3 2 0 0	0
0 3 2 0 1	0
0 5 0 0 0	1
0 5 1 0 0	0
0 5 1 2 0	6
0 6 0 0 1	1
0 7 0 1 6	6
0 7 7 0 3	1
1 0 0 2 0	1
1 0 2 2 1	3
1 0 7 1 0	3
1 1 1 6 1	2
1 1 7 2 3	2
1 2 2 0 0	3
1 2 3 3 1	2
1 3 2 0 0	3
1 3 3 2 1	2
1 3 5 1 0	3
1 6 0 2 0	5
1 6 3 0 1	1
1 7 1 1 0	3
2 0 1 0 7	2
2 0 7 1 0	3
2 1 0 2 0	5

N W C E S	new
2 1 2 0 0	1
2 1 3 1 2	2
2 2 0 0 0	2
2 2 2 0 0	1
2 3 3 1 0	1
3 1 0 1 3	1
3 1 1 1 7	3
3 1 3 5 7	1
3 2 7 1 3	2
3 7 0 0 0	1
3 7 5 2 0	1
5 0 0 0 0	2
5 0 7 0 7	0
5 3 0 2 0	1
5 7 0 0 0	1
6 0 0 0 0	7
6 1 0 0 0	1
6 1 6 3 1	2
6 1 7 1 2	3
6 1 7 1 7	3
7 0 0 0 0	7
7 0 7 0 0	0
7 1 0 0 0	1
7 1 3 1 2	2
7 1 3 6 1	1
7 2 3 3 0	7
7 2 7 3 0	2
7 3 3 0 0	0

(b)

Fig. 5. Transition function for the replication of the round loop from Fig. 6. (a) CMR representation discovered by the EA. (b) Corresponding table-based representation.

A. Results for the Round Loop

Although the round loop itself consists of only four different states [including the "empty" state 0—see Fig. 4(a)], the target CA works with eight cell states. This setup was chosen in order to increase the probability of discovering various replication scenarios, which was the main goal of these experiments. For this experiment, a minimum of four replicas were required to emerge (potentially including the initial loop) within a maximal number of 30 development steps. Note that during the evolution the initial loop was initialized in an inner part of the CA [as shown in Fig. 4(d)] in order to allow the EA to discover replication in any direction. This setup is motivated by some of the existing replicating loops that work in a similar way (i.e., the replicas can "grow" in more directions simultaneously).

The transition function of one of the best evolved solutions for the round loop is shown in Fig. 5, and the CA development performing the replication process is illustrated in Fig. 6. This CA represents the best result of this paper and will be referred to as Bidlo's loop. As evident from Fig. 6, the first replica is created after the 12th step on the right (east) from the initial loop. However, some cells are emerging both in the east and south before the first replica is finished. These cells actually represent a basis for the next replicas being created concurrently during the subsequent development. From this process it can be seen that the evolution discovered a new replication scheme—let us call it a diagonal replication as it creates successive copies of the loop diagonally between the east and south directions. The following stages can be identified with respect to the overall CA development: 1) the first replica emerges after the 12th step on the east side; 2) this replica produces the first diagonal level after the next eight steps; and 3) every next diagonal level is finished after six steps. Note that the development pattern of the CA is regular since stage 3, i.e., the number of replicas can be predicted and an analysis of the global CA behavior can be performed.

For comparison purposes, Byl's loop was chosen [17] because of its closest similarity to the round loop with respect to the shape, size, and complexity (see the illustration of the loops in Fig. 7). Both these loops were analyzed in detail, using our CA simulator, with the following results. The transition function of Byl's loop consists of 238 rules, our solution for the round loop works with 84 rules. The CA simulator can provide integer sequences containing the numbers of replicas at the respective time (i.e., after executing the CA for the appropriate number of steps). For example, the CA with Bidlo's loop (discovered in this paper) can produce $3, 6, 10, 15, 21, 28, 36, 45, \ldots$ replicas in $20, 26, 32, 38, 44, 50, 56, 62, \ldots$ steps, respectively. In order to mathematically express the number of Bidlo's loop depending on the number of CA steps N, a manual analysis of these sequences was performed; the result is described by

$$C_{\text{Bidlo}} = \frac{N^2 - 10N + 16}{72}; N \geq 20 \wedge (N - 20)\ \%\ 6 = 0. \quad (1)$$

A similar analysis was performed to express the number of Byl's loop as described by

$$C_{\text{Byl}} = \frac{2N^2}{625} - \frac{8N}{25} + 15; N \geq 100 \wedge N\ \%\ 25 = 0. \quad (2)$$

Note that the sign % in the formulas denotes the modulo division. Additional conditions for N are specified in order to express only the numbers of steps after which a group of replicas has just been completed (for other values of N the result is not a whole number). The correctness of the equations was confirmed using our CA simulator. Alternatively, suitable mathematical software can be

Appendix IV

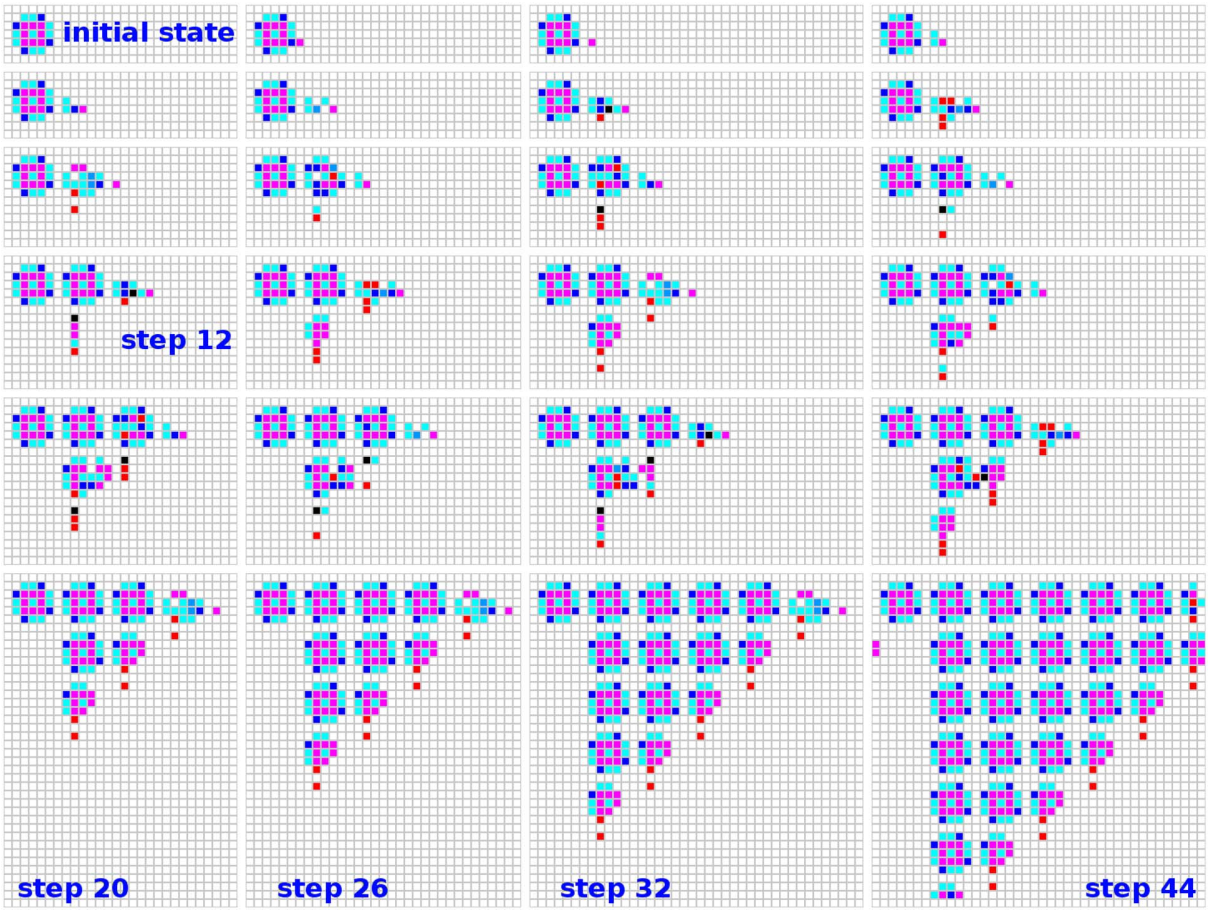


Fig. 6. Replication of the round loop in a CA developing according to the transition function from Fig. 5. The sequence of CA states reads from left to right, top to bottom.

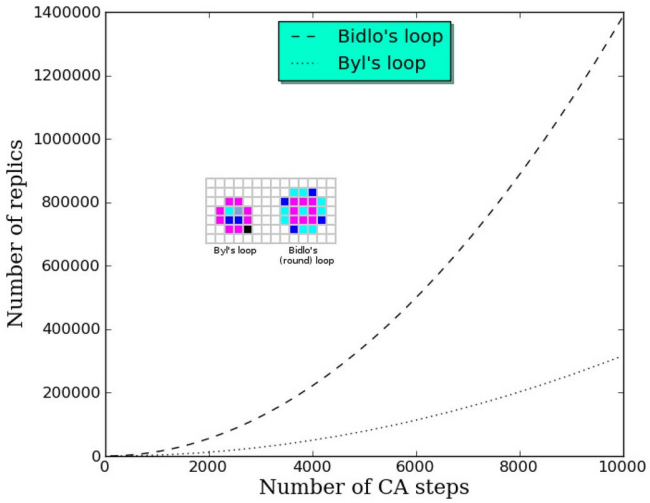


Fig. 7. Comparison of the number of replicas based on the number of CA steps for Byl's loop [17] and Bidlo's (round) loop.

involved. For example, WolframAlpha[2] represents a straightforward tool for such an analysis. By entering a part of the sequence containing the number of replicas for Byl's loop (15, 25, 39, 57, 79, 105, 135, 169), WolframAlpha provides a mathematical expression for this sequence as shown by

$$C_{\text{Byl}-\text{WA}} = 2x^2 + 4x + 9; x = 1, 2, 3, 4, 5, 6, \ldots \quad (3)$$

Note that in this case x represents an independent variable, not the number of the CA steps. Similarly, the number of replicas of Bidlo's loop can be expressed independently of N as shown, which was derived by WolframAlpha

$$C_{\text{Bidlo}-\text{WA}} = \frac{1}{2}(x+1)(x+2); x = 1, 2, 3, 4, 5, 6, \ldots \quad (4)$$

Again, our software simulator was applied that confirmed the correctness of the equations provided by WolframAlpha with respect to the output from the corresponding CA.

In the case of Byl's loop a group of replicas is completed after every 25 steps, for the round loop a whole diagonal of replicas is finished after every six steps. Although Byl's loop replicates into four directions, its replication speed is significantly lower in comparison with the round loop, which replicates in two dimensions only (see Fig. 7). This feature follows from the significantly lower factor of the element N^2 in (2) compared to (1) for the round loop. This means that the solution obtained in this paper is significantly more efficient (from the point of view of both the replication speed and the complexity of the transition function) compared to Byl's loop.

[2]http://www.wolframalpha.com/

Appendix IV

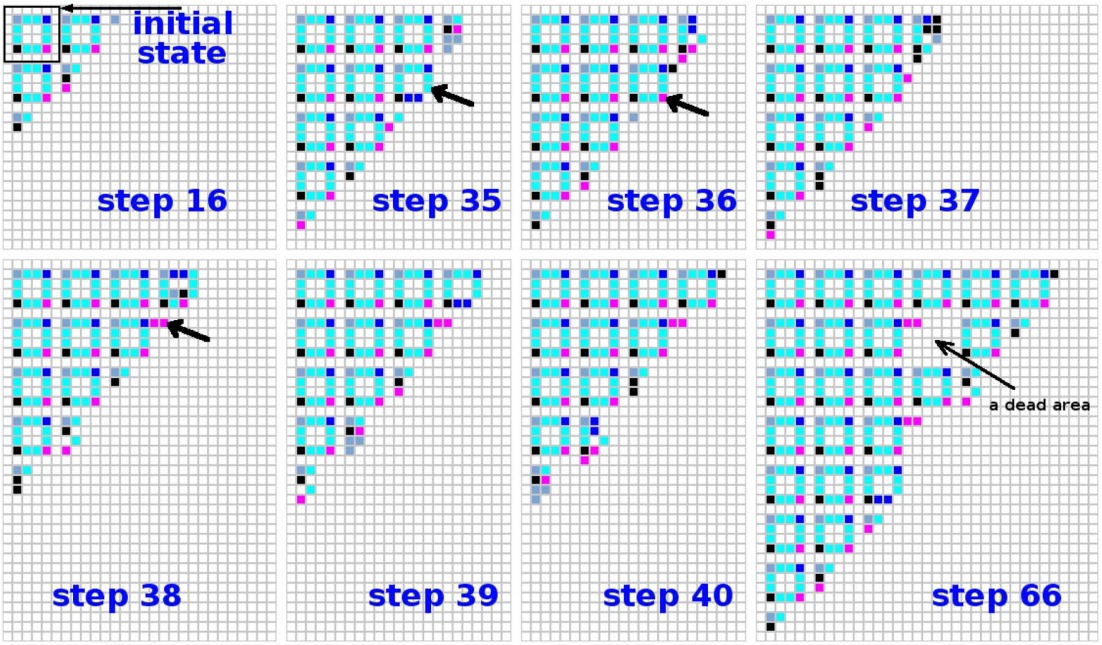


Fig. 8. Example of development of the rectangular loop discovered by the evolution. The CA works with eight states and the sequence of steps reads from left to right, top to bottom. The initial state is marked by a thick rectangle (top left).

Considering the replication period (i.e., the number of steps after which a new group of replicas is completed), which determines the replication speed, our solution even overcomes (with its replication period of six steps) all the best known replicating loops. For example, in addition to Byl's loop [17] with a replication period of 25 steps, Chou–Reggia's loop [18] has a 15-step replication period, Langton's loop [2] replicates in every 151 steps or Perrier's loop [60] exhibits a replication period of 235 steps. Therefore, Bidlo's (round) loop can be considered as the fastest replicating loop known so far and represents the main result of this paper.

B. Results for the Rectangular Loop

In order to show that the CMR encoding of the transition function is not limited to a specific loop only, other structures were also considered. The rectangular loop from Fig. 4(b) uses the same principle of the fitness evaluation as the round loop does. Several perfect solutions were obtained in which the aforementioned concept of diagonal replication can also be observed. One of those solutions is depicted in Fig. 8.

In this case, the development starts by a stage 1 that takes 16 steps during which two different replication processes are performed in both the east and south direction (the result of this stage is shown in the top-left part of Fig. 8). The next groups of replicas are developed during stage 2 after every 13 steps. However, it can be observed that the replication into the two directions is not symmetric [see Fig. 8, step 35 (the first row of replicas contains three complete loops whilst the south direction contains four loops in the first column)]. This irregularity causes that some loops are not able to finish their replication, which leads to some "dead areas," which emerge during the CA development. Since step 35 it is evident that the loop marked by the black arrow starts its east replication in step 36 but the arm constructed in step 38 (consisting of two cells in state 3) no longer develops during the next steps, producing a dead area that can be clearly visible after step 66. In order to identify the consequences of this anomaly a 200 × 200-cell CA was considered with the initial loop in its top-left corner. The development takes 871 steps until the CA reaches its stable state with the whole cellular space filled by replicas and dead areas (see Fig. 9), whose arrangement seems rather

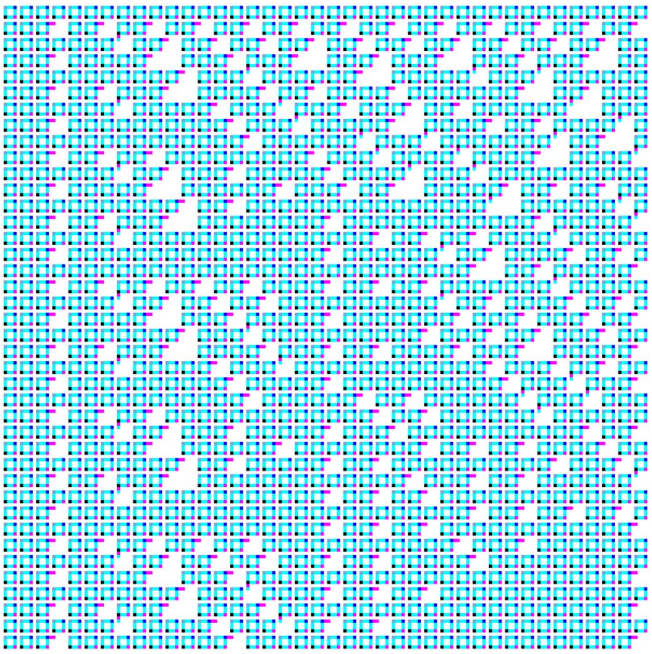


Fig. 9. Sample state of a 200×200-cell CA performing a replication of the rectangular loop. The state represents a continuation of the development from Fig. 8 after 871 steps and shows a chaotic arrangement of dead areas caused by an asymmetric diagonal replication.

Appendix IV

chaotic. Moreover, the development of this loop was observed using 400×400- and 800×800-cell CA with no systematic arrangement of the loops and dead areas that would allow predicting reliably the global behavior of such a CA.

Although the chaotic behavior is quite common for CA in class 3 of Wolfram's classification [23], it has been rarely observed in the case of replicating structures. However, the issue of prediction and potential applications of such behavior still represent open problems for the future research. Note that some other solutions were obtained that are able to replicate the rectangular loop without malformations, using the diagonal replication scheme.

C. Results for the Envelope Loop

The last experiment from the area of replicating structures demonstrates another approach to the fitness evaluation and considers a more complex replicating loop denominated as envelope loop [see Fig. 4(c)]. In this case the CA works with eight different cell states, the initial state consists of a single instance of the loop to be replicated (see the top-left state of Fig. 10 denoted as "initial state"). Contrary to the previous experiments, this replication task is evaluated using a fixed (reference) pattern as a complete CA state containing a copy of the original loop in a specific position (see the bottom-right state of Fig. 10 denoted as "step 20"). The reason for this experiment is to determine the ability of the EA to evolve CMRs for the transformation into an exact copy arrangement and to identify whether general replicators are possible in this case (i.e., the ability of the CA evolved to repeatedly generate replicas according to the original specification if the development continues). Since the fitness evaluation is different, compared to the previous loops, the success of this experiment also indicates that the CMR approach is robust.

The step fitness function is calculated after each step of the CA as the number of cells in correct states with respect to the reference pattern. The final fitness is defined as the maximum out of all the step fitness values. From a more general point of view the evaluation of the candidate solutions may be considered as a pattern transformation problem transforming the initial state into another target state containing the replica. The results are finally verified in order to determine whether the transition function is able to generate more replicas if the CA development continues. Fig. 10 shows an example of a successful solution performing a complete development of a single copy of the envelope loop. This solution is able to generate further replicas from the most recent one whereas each new replica is "shifted" by two cells down from its predecessor (as originally specified by the reference pattern during the evolution). The new replica is finished after 20 development steps and this solution represents the fastest replicator out of the results obtained for the envelope loop. The transition function consists of 102 table-based rules (transformed from the evolved CMR encoding). Note that various perfect results were obtained, for example, a CA working with 88 rules (the most compact transition function so far obtained for this loop) generating replicas in 25 steps, another solution uses 96 rules and the replication takes 30 steps.

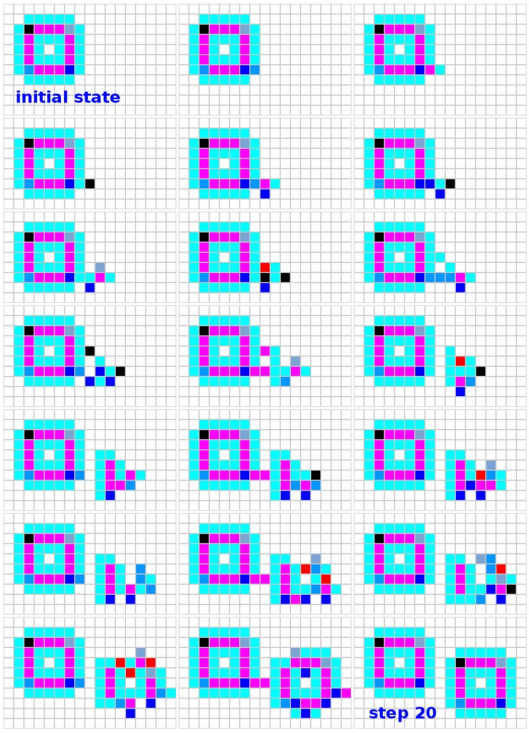


Fig. 10. CA development performing the replication of the envelope loop. The sequence of CA states reads from left to right, top to bottom.

D. Discussion

The EA provided working solutions that satisfy the given requirements in all the experiments, which indicates that the CMR approach is robust. Since the fitness function represents one of the key aspects of a successful evolutionary system (together with the representation of candidate solutions), it may enable the CMR concept to be successfully applied in advanced CA models in various areas (e.g., nonuniform, asynchronous, or probabilistic CA).

An observation of the evolved CA development showed that different replication techniques could be obtained even for a specific loop. For example, in addition to the results proposed in Section IV-A, a CA working with six cell states was discovered that needed the same number of steps (12) to replicate the initial loop but its transition function contained 99 rules (the solution from Section IV-A works with eight cell states and 84 rules, Byl's loop replicates according to 238 rules). In the case of both the round loop and the rectangular loop the replication process designed by the EA exhibits a pipelining principle (i.e., the replication of the next group of loops is in progress before the previous group has been finished). Although the replication process works for a specific direction only (e.g., it does not allow an easy modification or adaptation of the transition rules by "rotating" the cellular neighborhood known from Byl's loop [17]), it has been shown that the proposed loops replicate with a higher speed against the existing loops. Another important aspect discovered by the EA, which probably contributes to the replication speed, is the diagonal replication. Moreover, it seems that the diagonal replication represents a technique that enabled reducing

Appendix IV

the number of transition rules needed to perform the replication. In particular, the proposed Bidlo's loop can create 528 instances in 200 CA steps compared to 79 instances of Byl's loop in the same time (i.e., in this case the proposed loop produced more than six times more copies than Byl's loop).

Although the form or arrangement of the proposed loops was inspired by the existing (self-replicating) loops, the results obtained do not exhibit the concept of self-replication in which the information needed to create the replica is encoded in the loop body. In fact, no exact solution to any existing self-replicating loop has yet been rediscovered using the proposed approach. The information encoded in the loop, which specifies the self-replication features, determines the CA development, which is specific to the given loop (e.g., Langton's loop encodes a sequence of states that determine how to create the replication "arm," when to turn the arm in order to create the corners of the loop or how to "close" the final replica). The transition function must interpret this information in order to control the CA development accordingly. The problem is that if such a transition function is specific to a given loop (i.e., there are no or only very few transition functions in the solution space that would perform self-replication of the given loop), then the EA may not be able to find the solution in a reasonable time.

However, the proposed approach allowed discovering transformations for creating replicas that are completely new or even exhibit some phenomena that would probably not be intended or acceptable during a manual CA design. In order to enable the EA to design innovations, it is important for the evaluation of the candidate solutions not to be very strict. In particular, the diagonal replication scheme, pipelining principle or malformation of the loops during the replication process represent phenomena designed by the EA that were not explicitly specified (required) by the designer.

V. Conclusion

In this paper, some selected applications of CMRs were presented for a routine evolutionary design of complex multistate uniform CA. The proposed EA in combination with the CMR encoding was able to find working solutions to problems for which the conventional representation techniques failed. In particular, a novel replication scheme in 2-D CA was discovered in this paper that allows a faster development of the copies of the given structure in comparison with the known approaches. Some techniques were evolved to calculate the square of integer numbers in 1-D CA that require a considerably lower number of steps compared to the existing solution. The results obtained represent the first case of a successful automatic design of multistate CA for this kind of problems.

In general, the proposed approach showed the ability to find working solutions to various kinds of problems (the square calculation in 1-D CA and replication in 2-D CA, where the result can be specified either as a minimal number of replicas or by a fixed pattern). This indicates that other classes of problems could be successfully solved in the future. For example, the CMRs might allow optimizing the pattern generation in the area of computer graphics. Other potential applications may include the design of CA for random number generation or the optimization of test pattern generators for digital circuits.

As regards the proposed case studies, the results bring some open questions whose investigation could be beneficial to both elementary and advanced CA-based models. For instance, what is the best way of encoding the information on self-replication for the purposes of the evolutionary design? Could the CMRs be adapted in order to provide the capability to optimize the number of states during evolution? Are there any new operators for the EA that would allow optimizing the evolutionary process itself? These issues represent ideas for our future research.

References

[1] J. Von Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, Ed. Urbana, IL, USA: Univ. Illinois Press, 1966.

[2] C. G. Langton, "Studying artificial life with cellular automata," *Phys. D Nonlin. Phenom.*, vol. 22, nos. 1–3, pp. 120–149, 1986.

[3] M. Sipper, "Studying artificial life using a simple, general cellular model," *Artif. Life*, vol. 2, no. 1, pp. 1–35, 1994.

[4] F. Peper, J. Lee, S. Adachi, and T. Isokawa, "Cellular nanocomputers: A focused review," *Int. J. Nanotechnol. Mol. Comput.*, vol. 1, no. 1, pp. 33–49, 2009.

[5] P. L. Rosin, A. Adamatzky, and X. Sun, *Cellular Automata in Image Processing and Geometry*. Cham, Switzerland: Springer, 2014.

[6] M. Espinola, J. A. Piedra-Fernandez, R. Ayala, L. Iribarne, and J. Z. Wang, "Contextual and hierarchical classification of satellite images based on cellular automata," *IEEE Geosci. Remote Sens.*, vol. 53, no. 2, pp. 795–809, Feb. 2015.

[7] S. Sahu *et al.*, "On cellular automata rules of molecular arrays," *Natural Comput.*, vol. 11, no. 2, pp. 311–321, 2012.

[8] P. Yin, S. Sahu, A. J. Turberfield, and J. H. Reif, "Design of autonomous DNA cellular automata," in *DNA Computing* (LNCS 3892), A. Carbone and N. A. Pierce, Eds. Heidelberg, Germany: Springer, 2006, pp. 399–416.

[9] G. C. Sirakoulis, "Parallel application of hybrid DNA cellular automata for pseudorandom number generation," *J. Cell. Autom.*, vol. 11, no. 1, pp. 63–89, 2016.

[10] E. F. Codd, *Cellular Automata*. New York, NY, USA: Academic, 1968.

[11] M. Sipper, "Quasi-uniform computation-universal cellular automata," in *Advances in Artificial Life* (LNCS 929). Heidelberg, Germany: Springer, 1995, pp. 544–554.

[12] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays*, vol. 4, 2nd ed. Natick, MA, USA: CRC, 2004.

[13] J.-B. Yunés, "Achieving universal computations on one-dimensional cellular automata," in *Cellular Automata for Research and Industry* (LNCS 6350). Heidelberg, Germany: Springer, 2010, pp. 660–669.

[14] G. D. Stefano and A. Navarra, "Scintillae: How to approach computing systems by means of cellular automata," in *Cellular Automata for Research and Industry* (LNCS 7495). Heidelberg, Germany: Springer, 2012, pp. 534–543.

[15] P. Rendell, "A fully universal turing machine in Conway's game of life," *J. Cell. Autom.*, vol. 8, nos. 1–2, pp. 19–38, 2013.

[16] C. G. Langton, "Self-reproduction in cellular automata," *Phys. D Nonlin. Phenom.*, vol. 10, nos. 1–2, pp. 135–144, 1984.

[17] J. Byl, "Self-reproduction in small cellular automata," *Phys. D Nonlin. Phenom.*, vol. 34, nos. 1–2, pp. 295–299, 1989.

[18] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng, "Simple systems that exhibit self-directed replication," *Science*, vol. 259, no. 5099, pp. 1282–1287, 1993.

[19] G. Tempesti, "A new self-reproducing cellular automaton capable of construction and computation," in *Proc. 3rd Eur. Conf. Artif. Life*, vol. 929. Granada, Spain, 1995, pp. 555–563.

[20] M. Land and R. K. Belew, "No perfect two-state cellular automata for density classification exists," *Phys. Rev. Lett.*, vol. 74, no. 25, pp. 5148–5150, 1995.

[21] M. S. Capcarrere, M. Sipper, and M. Tomassini, "Two-state, $r = 1$ cellular automaton that classifies density," *Phys. Rev. Lett.*, vol. 77, pp. 4969–4971, Dec. 1996.

Appendix IV

[22] S. Sahoo, P. P. Choudhury, A. Pal, and B. K. Nayak, "Solutions on 1-D and 2-D density classification problem using programmable cellular automata," *J. Cell. Autom.*, vol. 9, no. 1, pp. 59–88, 2014.

[23] S. Wolfram, *A New Kind of Science*. Champaign, IL, USA: Wolfram Media, 2002.

[24] S. Ninagawa, "Solving the parity problem with rule 60 in array size of the power of two," *J. Cell. Autom.*, vol. 8, nos. 3–4, pp. 189–203, 2013.

[25] A. Adamatzky, *Identification of Cellular Automata*. Boca Raton, FL, USA: CRC, 1994.

[26] L. Bull, I. Lawson, A. Adamatzky, and B. DeLacyCostello, "Towards predicting spatial complexity: A learning classifier system approach to the identification of cellular automata," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Edinburgh, U.K., 2005, pp. 136–141.

[27] A. Adamatzky, "Identification of cellular automata," in *Computational Complexity*, R. A. Meyers, Ed. New York, NY, USA: Springer, 2012, pp. 1564–1575.

[28] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[29] N. H. Packard, "Adaptation toward the edge of chaos," in *Dynamic Patterns in Complex Systems*, J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, Eds. Teaneck, NJ, USA: World Sci., 1988, pp. 293–301.

[30] F. C. Richards, T. P. Meyer, and N. H. Packard, "Extracting cellular automaton rules directly from experimental data," *Phys. D Nonlin. Phenom.*, vol. 45, nos. 1–3, pp. 189–202, 1990.

[31] E. Sapin, O. Bailleux, and J. Chabrier, "Research of complexity in cellular automata through evolutionary algorithms," *Complex Syst.*, vol. 17, no. 3, pp. 231–241, 2007.

[32] M. Sipper, *Evolution of Parallel Cellular Machines—The Cellular Programming Approach* (LNCS 1194). Heidelberg, Germany: Springer, 1997.

[33] R. Breukelaar and T. Bäck, "Using a genetic algorithm to evolve behavior in multi dimensional cellular automata," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Washington, DC, USA, 2005, pp. 107–114.

[34] M. Mitchell, J. P. Crutchfield, and P. T. Hraber, "Evolving cellular automata to perform computations: Mechanisms and impediments," *Phys. D*, vol. 75, nos. 1–3, pp. 361–391, 1994.

[35] E. Sapin, "Gliders and glider guns discovery in cellular automata," in *Game of Life Cellular Automata*, A. Adamatzky, Ed. London, U.K.: Springer, 2010, pp. 135–165.

[36] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. New York, NY, USA: Wiley, 1966.

[37] W. Elmenreich and I. Fehérvári, "Evolving self-organizing cellular automata based on neural network genotypes," in *Proc. 5th Int. Conf. Self-Organizing Syst.*, Karlsruhe, Germany, 2011, pp. 16–25.

[38] D. Medernach, T. Kowaliw, C. Ryan, and R. Doursat, "Long-term evolutionary dynamics in heterogeneous cellular automata," in *Proc. 15th Annu. Conf. Genet. Evol. Comput.*, Amsterdam, The Netherlands, 2013, pp. 231–238.

[39] S. Bandini, A. Bonomi, and G. Vizzari, "An analysis of different types and effects of asynchronicity in cellular automata update schemes," *Nat. Comput.*, vol. 11, no. 2, pp. 277–287, 2012.

[40] M. A. J. Javid, M. M. al-Rifaie, and R. Zimmer, "Detecting symmetry in cellular automata generated patterns using swarm intelligence," in *Theory and Practice of Natural Computing* (LNCS 8890), A.-H. Dediu, M. Lozano, and C. Martín-Vide, Eds. Cham, Switzerland: Springer, 2014, pp. 83–94.

[41] J. Skaruz, F. Seredynski, and A. Piwonska, "Two-dimensional patterns and images reconstruction with use of cellular automata," *J. Supercomput.*, vol. 69, no. 1, pp. 9–16, 2014.

[42] J. M. Baetens and B. De Baets, "Towards a comprehensive understanding of multi-state cellular automata," in *Cellular Automata* (LNCS 8751), J. Wąs, G. C. Sirakoulis, and S. Bandini, Eds. Cham, Switzerland: Springer, 2014, pp. 16–24.

[43] J. M. Baetens and B. De Baets, "Phenomenological study of irregular cellular automata based on Lyapunov exponents and Jacobians," *Chaos*, vol. 20, no. 3, 2010, Art. ID 033112.

[44] A. Bisio, G. M. D'Ariano, P. Perinotti, and A. Tosini, "Free quantum field theory from quantum cellular automata," *Found. Phys.*, vol. 45, no. 10, pp. 1137–1152, 2015.

[45] V. A. Mardiris, G. C. Sirakoulis, and I. G. Karafyllidis, "Automated design architecture for 1-D cellular automata using quantum cellular automata," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2476–2489, Sep. 2015.

[46] K. Sridharan and V. Pudi, *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology*. Cham, Switzerland: Springer, 2015.

[47] A. N. Bahar, S. Waheed, and N. Hossain, "A new approach of presenting reversible logic gate in nanoscale," *SpringerOpen J. Electron. Electr. Eng.*, vol. 4, no. 1, p. 153, 2015.

[48] S. Sahu *et al.*, "Molecular implementations of cellular automata," in *Cellular Automata for Research and Industry* (LNCS 6350). Heidelberg, Germany: Springer, 2010, pp. 650–659.

[49] H. Balijepalli and M. Niamat, "Design of a nanoscale quantum-dot cellular automata configurable logic block for FPGAs," in *Proc. IEEE 55th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Boise, ID, USA, Aug. 2012, pp. 622–625.

[50] M.-A. I. Tsompanas, G. C. Sirakoulis, and A. I. Adamatzky, "Evolving transport networks with cellular automata models inspired by slime mould," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1887–1899, Sep. 2015.

[51] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[52] D. Andre, F. H. Bennett, III, and J. R. Koza, "Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem," in *Proc. 1st Annu. Conf. Genet. Evol. Comput.*, 1996, pp. 3–11.

[53] D. Andre, F. H. Bennett, III, and J. R. Koza, "Evolution of intricate long-distance communication signals in cellular automata using genetic programming," in *Proc. Artif. Life V 5th Int. Workshop Synth. Simulat. Living Syst.*, 1996, pp. 513–522.

[54] Y. Zhao and S. A. Billings, "The identification of cellular automata," *J. Cellular Automata*, vol. 2, no. 1, pp. 47–65, 2007.

[55] K.-I. Maeda and C. Sakama, "Identifying cellular automata rules," *J. Cellular Automata*, vol. 2, no. 1, pp. 1–20, 2007.

[56] M. F. Brameier and W. Banzhaf, *Linear Genetic Programming*. New York, NY, USA: Springer, 2007.

[57] M. Bidlo and Z. Vasicek, "Evolution of cellular automata using instruction-based approach," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1060–1067.

[58] M. Bidlo and Z. Vasicek, "Evolution of cellular automata with conditionally matching rules," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Cancún, Mexico, 2013, pp. 1178–1185.

[59] M. Bidlo, "Evolving multiplication as emergent behavior in cellular automata using conditionally matching rules," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 2732–2739.

[60] J.-Y. Perrier, M. Sipper, and J. Zahnd, "Toward a viable, self-reproducing universal computer," *Phys. D Nonlin. Phenom.*, vol. 97, no. 4, pp. 335–352, 1996.

Michal Bidlo received the Ph.D. degree in information technology from the Faculty of Information Technology (FIT), Brno University of Technology (BUT), Brno, Czech Republic, in 2009.

He is an Assistant Professor with the Department of Computer Systems, FIT, BUT. He has authored or co-authored over 20 conference/journal papers focused on evolutionary design and evolvable hardware. His research interests include cellular automata, evolutionary computation, evolvable hardware, and bio-inspired systems.

Appendix V

Evolution of Cellular Automata-Based Replicating Structures Exhibiting Unconventional Features

BIDLO Michal

In: *International Joint Conference on Computational Intelligence, IJCCI 2015, Lisbon, Portugal, November 12-14, 2015, Revised Selected Papers.* Cham: Springer International Publishing, 2016, pp. 21-41. ISBN 978-3-319-48506-5.

The original work was published in [19], 2015 International Joint Conference on Computational Intelligence (IJCCI), Lisbon, Portugal, and selected for publication of an extended version hereafter.

Evolution of Cellular Automata-Based Replicating Structures Exhibiting Unconventional Features

Michal Bidlo

Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, 61266 Brno, Czech Republic
E-mail: bidlom@fit.vutbr.cz
http://www.fit.vutbr.cz/~bidlom

Abstract. Replicating loops represent a class of benchmarks, which is commonly studied in relation with cellular automata. Most of the known loops, for which replication rules exist in two-dimensional cellular space, create the copies of themselves using a certain construction algorithm that is common for all the emerging replicas. In such cases, the replication starts from a single instance of the loop (represented as the initial state of the cellular automaton) and is controlled by the transition function of the automaton according to which the copies of the loop are developed. Despite the fact that universal replicators in cellular automata are possible (for example, von Neumann's Universal Constructor), the process of replication of the loops is usually specific to the shape of the loop and the replication rules given by the transition function. This work presents a method for the automatic evolutionary design of cellular automata, which allows us to design transition functions for various structures that are able to replicate according to a given specification. It will be shown that new replicating loops can be discovered that exhibit some unconventional features in comparison with the known solutions. In particular, several scenarios will be presented which can, in addition to the replication from the initial loop, autonomously develop the given loop from a seed, with the ability of the loop to subsequently produce its replicas according to the given specification. Moreover, a parallel replicator will be shown that is able to develop the replicas to several directions using different replication algorithms.

Keywords: genetic algorithm, cellular automaton, transition function, conditional rule, replicating loop

1 Introduction

Since the introduction of cellular automata (CA) in [von Neumann, 1966], researchers have dealt, among others, how to effectively design a cellular automaton (and its transition function in particular) to solve various problems.

2

For example, cellular automata have been studied for their ability to perform computations, e.g. using principles from the famous Conway's Game of Life [Berlekamp et al., 2004] or by simulating elementary logic functions in non-uniform cellular matrix [Sipper, 1995].

One of the topics widely studied in the area of artificial life is the problem of (self-)replicating loops. Since the introduction of probably the most known loop by Langton [Langton, 1984], which is able to replicate in 151 steps in a CA working with 8 states, some other researchers have dealt with this topic trying to simplify the replication process or enhance the abilities of the loop during replication. For example, Byl introduced a smaller loop that is able to replicate in 25 steps using a CA that works with 6 cell states [Byl, 1989]. Later, several unsheathed loops were proposed by Reggia et al. from which the simplest loop consists of 6 cells only and is able to replicate using 8-state CA in 14 steps [Reggia et al., 1993]. On the other hand, Tempesti studied a possibility to introduce construction capabilities into the loops and proposed a 10-state CA that allows to generate patters inside the replicating structures [Tempesti, 1995]. Perrier et al. created a "self-reproducing universal computer" using 64-state CA by "attaching" executable programs (Turing Machines) on the loops [Perrier et al., 1996]. Although the aforementioned solutions were achieved using analytic methods, the process of determining suitable transition rules for a given problem represents a difficult task and requires an experienced designer (the process of "programming" the CA is not intuitive). As the number of cell states increases, the process of the CA design becomes challenging due to a significant increase of the solution space. Moreover, for some problems no analytic approach has yet been known to the design of the transition rules. In such cases various unconventional techniques have been applied including Genetic Algorithm (GA) [Holland, 1975]), possibly in combination with other heuristics.

For example, Mitchell et al. investigated a problem of performing computations in cellular automata using GA [Mitchell et al., 1993]. Their work contains a comparison with the original results obtained by Packard in [Packard, 1988] which can be considered as a milestone in applying evolutionary algorithms (EA) to the design and optimisation of cellular automata. In particular, the authors in [Mitchell et al., 1993] claim: "Our experiment produced quite different results, and we suggest that the interpretation of the original results is not correct." It may indicate that the research of cellular automata (and their typical features like emergent behaviour or cooperative cell signalling by means of local rules) using various computing techniques can provide valuable information for advanced studies and applications in this area. Note that Mitchell et al. considered binary (i.e. 2-state) 1D cellular automata only which represent a fundamental concept for advanced models. Sipper proposed a technique called Cellular Programming (a spatially distributed and locally interacting GA) that allows for the automatic design of non-uniform CA that are well suited to various problems [Sipper, 1997]. Sapin et al. introduced a GA-based approach to the design of gliders and glider guns in 2D cellular automata [Sapin and Bull, 2008][Sapin et al., 2010]. It was shown that a sponta-

neous emergence of glider guns in CA can occur with a significant number of new gun-based and glider structures discovered by EA. The aim of the glider research was to construct a system for collision-based computationally universal cellular automata that are able to simulate Turing machines [Sapin and Bull, 2008]. In recent years, several solutions emerged that aim to optimize the CA design by introducing various evolution-based and soft-computing techniques in combination with suitable representations of the transition functions. For example, Elmenreich et al. proposed an original technique for the calculation of the transition function using neural networks (NN) [Elmenreich and Fehérvári, 2011]. The goal was to train the NN by means of Evolutionary Programming [Fogel et al., 1966] in order to develop self-organising structures in the CA. A novel technique for encoding the transition functions of CA, called Conditionally Matching Rules, was introduced in [Bidlo and Vasicek, 2013], and some applications in binary CA with advantages over the conventional (table-based) encoding were presented in [Bidlo, 2014].

Whilst the most of the aforementioned studies considered binary CA (i.e. those working with two cell states only), which may be suitable for straightforward hardware implementations (e.g. Sipper's Firefly machine [Sipper et al., 1997]), multi-state CA can provide a more efficient way for the representation and processing of information in CA thanks to the ability of individual cells to work with more than two states. This feature is important for studying complex systems that are in most cases described by integer (or real-valued) variables. In addition, the introduction of more than two states per cell in the CA may allow to reduce the resources needed to solve a given problem (e.g. the size of the cellular array or dimension of the automaton). For example, Yunès studied computational universality in multi-state one-dimensional cellular automata [Yunès, 2010]. A technique for the construction of computing systems in 2D CA was demonstrated by Stefano and Navarra in [Stefano and Navarra, 2012] using rules of a simple game called Scintillae working with 6 cell states. Their approach allows to design components (building blocks) for the construction of bigger systems, e.g. on the basis of gate-level circuits.

The goal of this study is to demonstrate the evolutionary design of 2D cellular automata, using the concept of conditionally matching rules to encode the transition functions, which are able to replicate the given structures with respect to a given arrangement in the cellular array. In particular, uniform, multi-state cellular automata will be treated, the cells of which work with 8 and 10 states. The GA will be applied in order to design suitable transition rules that perform replication of the given structure according to the designer's specification. It will be shown that novel replication scenarios can be found in CA that can copy the given loop not only from its initial instance but also, from a seed the loop can autonomously grow. Moreover, a parallel replication scheme will be presented, the objective of which is to speed-up the replication process by allowing the structures to replicate to more directions in the 2D CA. The results will demonstrate the ability of the GA to discover different replication scenarios for the

replicas developing in parallel in the cellular automaton, which will be encoded in a single evolved transition function.

2 Cellular Automata

The original concept of cellular automaton, introduced in [von Neumann, 1966], which will be considered in this study, assumes a 2D matrix of cells, each of which at a given moment acquires a state from a finite set of states. The development of the CA is performed synchronously in discrete iterations (time steps) by updating the cell states according to local transition functions of the cells. Uniform cellular automata will be investigated in which the local transition function is identical for all cells and hence it can be considered as a transition function of the CA. The next state of each cell depends on the combination of states in its neighbourhood. In this work, von Neumann neighbourhood will be assumed that includes a given (*C*entral) cell to be updated and its immediate neighbours in the *N*orth, *S*outh, *E*ast and *W*est direction (i.e. it is a case of a 5-cell neighbourhood).

Since the CA behaviour can practically be evaluated in the cellular array of a finite size, boundary conditions need to be specified in order to correctly determine cell states at the edge of the array. Cyclic boundary conditions will be implemented which means that cells at an edge of the CA are "connected" to the appropriate cells on the opposite edge (i.e. these cells are considered as neighbours) in each dimension. In case of the 2D CA the shape of such cellular array can be viewed as a toroid.

The transition function is usually defined as a mapping that for all possible combinations of states in the cellular neighbourhood determines a new state. This mapping can be represented as a set of rules of the form $N_t\ W_t\ C_t\ E_t\ S_t \rightarrow C_{t+1}$ where N_t, W_t, C_t, E_t and S_t denote cell states in the defined neighbourhood at a time t and C_{t+1} is the new state of the cell to be updated. It means that for every possible combination of states $N_t\ W_t\ C_t\ E_t\ S_t$ a new state C_{t+1} needs to be specified. However, if the number of cell states increases, the number of possible transition rules grows significantly which is inconvenient for efficient CA design. Of course, not all transition rules need to be specified explicitly but the problem is how to choose the rules which modify the central cell in the neighbourhood. Therefore, an advanced representation of the transition rules was proposed and denominated as Conditionally Matching Rules [Bidlo and Vasicek, 2013]. Conditionally matching rules allows us to reduce the size of representation of the transition functions especially with respect to the evolutionary design of cellular automata.

3 Conditionally Matching Rules

The concept of conditionally matching rules (CMR) showed as a very promising technique in comparison with the conventional (table-based) approach considering various experiments with binary cellular automata (e.g. pattern development

task [Bidlo and Vasicek, 2013] or binary multiplication in 2D CA [Bidlo, 2014]). In this work, evolutionary design of the CMR-based representation will be investigated in order to design cellular automata with up to 10 cell states that support replication of a given structure.

A conditionally matching rule represents a generalised rule of a transition function for determining a new cell state. Whilst the common approach specifies a new state for every given combination of states in the cellular neighbourhood, the CMR-based approach allows to encode a wider range of combinations into a single rule. A CMR is composed of two parts: a condition part and a new state. The number of items (size) of the condition part corresponds to the number of cells in the cellular neighbourhood. Let us define a condition item as an ordered pair consisting of a condition function and a state value. The condition function is typically expressed as a function whose result can be interpreted as either true or false. The condition function evaluates the state value in the condition item with respect to the state of the appropriate cell in the cellular neighbourhood. In particular, each item of the condition part is associated with a cell in the neighbourhood with respect to which the condition is evaluated. If the result of such evaluation is true, then the condition item is said to match with the state of the appropriate cell in the neighbourhood. In order to determine a new cell state according to a given CMR, all its condition items must match (in such case the CMR is said to match).

The following condition functions will be considered: $== 0, \neq 0, \leq, \geq$. Note that this condition set represents a result of our long-term experimentation and experience with the CMR approach and will be used for all the experiments in this study. The condition $== 0$, respective $\neq 0$, evaluates whether the corresponding cell state is equal to 0 (i.e. a "dead" state), respective whether it is different from state 0. Note that the state value of the condition item for $== 0$ and $\neq 0$ is considered implicitly within the condition itself. The conditions $\leq$ and $\geq$ represent relational operators "less or equal" and "greater or equal" respectively for which the state value of the condition item must be explicitly specified.

Figure 1 shows an example of conditionally matching rules defined for a 2D CA with the 5-cell neighbourhood together with the illustration of cells the condition items are related to. CMR (A) is a matching CMR since all the conditions of its condition part are evaluated as true with respect to the sample neighbourhood shown in the left part of Fig. 1. On the other hand, CMR (B) does not match because the second condition item $! = 2$ evaluates as false with respect to the west cell that possesses state 2. Similarly, the third condition $== 0$ of CMR (B) is not true as the central cell is in state 2.

A CMR-based transition function can be specified as a finite (ordered) sequence of conditionally matching rules. The following algorithm will be applied to determine a new state of a cell. The CMRs are evaluated sequentially one by one. The first matching CMR in the sequence is used to determine the new state. If no of the CMRs matches, then the cell keeps its current state. These conventions for evaluating and applying the CMRs ensure that the process of cal-

6

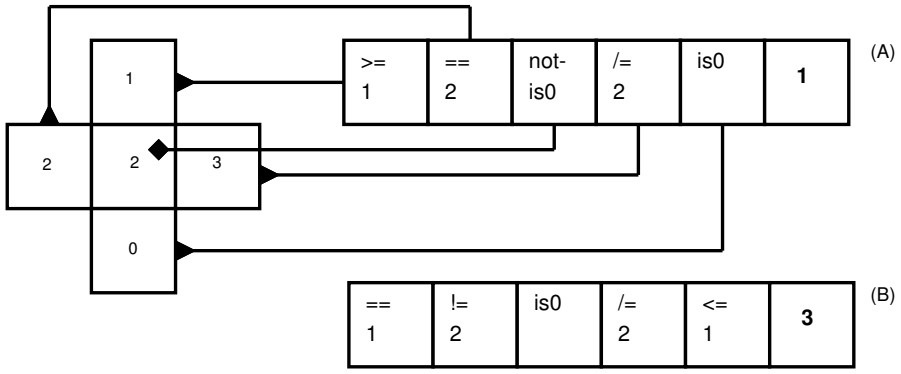


Fig. 1. Example of a conditionally matching rule specified for 5-cell neighbourhood. The value of the new state is written in bold. (A) example of a matching CMR, (B) example of a CMR that does not match – the second and third condition is evaluated as false.

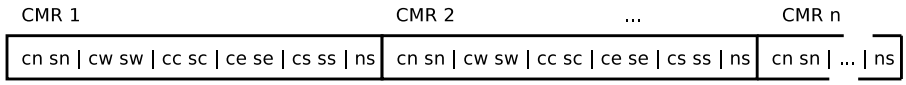


Fig. 2. Structure of a chromosome for genetic algorithm encoding a CMR-based transition function. cx denote a condition for the cell at position x in the neighbourhood, sx represents the state value to be investigated using the appropriate condition with respect to the state of cell at position x, ns specifies the next state for a given CMR. All the conditions and state values are represented by integer numbers.

culating the new state is deterministic (it is assumed that the condition functions are deterministic too). Therefore, it is possible to convert the CMR-based transition function to a corresponding table-based representation which preserves the fundamental concept of cellular automata. Moreover, every condition set that includes relation == allows to formulate transition rules for specific combinations of states if needed (by specifying == for all condition items of the CMR).

In order to obtain the conventional (table-based) representation of the transition rules from an evolved CMR-based solution, the following algorithm is applied using the same CA that was considered during evolution. Let C_t and C_{t+1} denote states of a cell in two successive steps of the CA at time t and $t+1$ respectively. A transition rule of the form $N_t\ W_t\ C_t\ E_t\ S_t \rightarrow C_{t+1}$ is generated for the combination of states in the cellular neighbourhood if $C_t \neq C_{t+1}$. This process is performed after each step and for each cell until the CA reaches a stable or periodic state. The set of rules obtained from this process represents the corresponding conventional prescription of the transition function. Note that only the rules that modify the cell state are generated, all the other rules are implicitly considered to preserve the current state.

4 Evolutionary System Setup

A genetic algorithm is utilized for the evolution of CMR-based transition functions in order to achieve the given behaviour in cellular automata. Each chromosome of the GA represents a candidate transition function encoded as a finite sequence of CMRs. The chromosome is implemented as a vector of integers in which the condition items and next states of the CMRs are encoded. Note that the population consists of chromosomes of a uniform length (given by the number of CMRs) which is specified as a parameter for a specific experiment. The structure of a chromosome is depicted in Figure 2.

The population of the GA consists of 8 chromosomes that are initialised randomly at the beginning of the evolutionary process. In each generation, four individuals are selected randomly from the current population, the best one of which is considered as a parent. In order to generate an offspring, the parent undergoes a process of mutation as follows. A random integer M in range from 0 to 2 is generated. Then M random positions in the parent chromosome are selected. The offspring is created by replacing the original integers at these positions by new valid randomly generated values. If M equals 0, then no mutation is performed and the offspring is identical to the parent. The process of selection and mutation is repeated until the entire new population is created. Crossover is not applied because no benefit of this operator was observed during the initial experiments. Note that the same GA has successfully been applied since the introduction of CMRs in various case studies [Bidlo and Vasicek, 2013][Bidlo, 2014]. Although no optimal (evolutionary) approach has yet been known for uniform CA, our experiments indicate that small-population EA (i.e. less than 10 individuals) with a simple mutation operator may represent a suitable class of algorithms to obtain working solutions with a reasonable success rate and computational effort. However, the detailed analysis and wider comparison of different techniques is not a subject of this study.

For each experiment, the GA is executed for 3 million generations. If no correct solution is found within this limit, the evolution is terminated. The evaluation of the chromosomes (i.e. the fitness function) and details regarding various experimental settings are described in the next section.

5 Experimental Results

This section summarises statistics of the evolutionary experiments performed and presents some results together with a more detailed analysis. Two sets of experiments are considered, the goal of each is to design CA that is able to replicate the given loop. The first set works with a *big loop* (the denomination is chosen for the purposes of this work with respect to the loop in the second set of experiments), the objective is to design transition rules that are able to develop a single replica of the loop in a given arrangement against the initial loop. In the second set, a simpler, *small loop* is treated, the goal is to find replication rules for the development of two independent replicas in parallel on the left and right

side of the initial loop. Note that the loops consist of cells in 7 different states (including state 0). In both sets of experiments, the CA working with 8 and 10 cell states are investigated. Moreover, different numbers of CMRs (varying from 20 to 50) encoded in the GA chromosomes are considered. For each setup, 100 independent evolutionary runs are executed. The experiments were executed using the Anselm cluster[1], the time of a single run (3 million generations) is approximately 12 hours.

5.1 Replication Evolution of the Big Loop

A big loop is considered for the replication in the first set of experiments, the structure of which is shown in Figure 3a. The genetic algorithm is applied to design the transition rules for the CA, which perform the replication of the loop in a maximum of 30 steps. The required CA state, that contains the replica, is depicted in Figure 3b. The following algorithm is applied to the evaluation of the candidate solutions during evolution and the calculation of the fitness function. A partial fitness function is evaluated after each CA step as the number of cells in correct states with respect to Figure 3b. The final fitness value of a given candidate solution is defined as the maximum of the partial fitness values. It this case the replication can be considered as a pattern transformation problem from a single (initial) loop onto two loops in a given arrangement. However, the loop is expected to replicate again and again during the subsequent CA development, which will be validated for the results obtained from the evolution. Moreover, an assumption is considered that each newly created loop is shifted by two cells down with respect to its predecessor (as shown in Fig. 3b). Therefore, the solutions obtained are further investigated using a visual software simulator developed by the author of this work in order to check that. The goal of this approach is to determine whether the GA is able to discover various new general replication scenarios. Note that, for the purposes of this study, the term "general" means the ability of a solution to repeatedly produce more replicas of the given loop, not an ability to replicate arbitrary loops.

Table 1 summarises the results of experiments with the big loop and provides an overview of some basic parameters of the CA that can be observed during its development using the evolved transition functions. As evident, the maximum success rate achieved during the experiments is only 12% which is not very high. Note, however, that the replication of the proposed loop represents a problem for which no working solution was found during our previous experiments using the table-based transition functions.

In addition to the results obtained for the CA working with 8 cell states, some successful solutions have even been obtained for 10 cell states which indicates that the CMRs are an efficient encoding of the transition rules that allows for the design of more complex multi-state CA. The solutions obtained in this work demonstrate a wide range of various replication schemes that can be performed using CA. For example, a solution was found that is able to replicate the loop

[1] https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

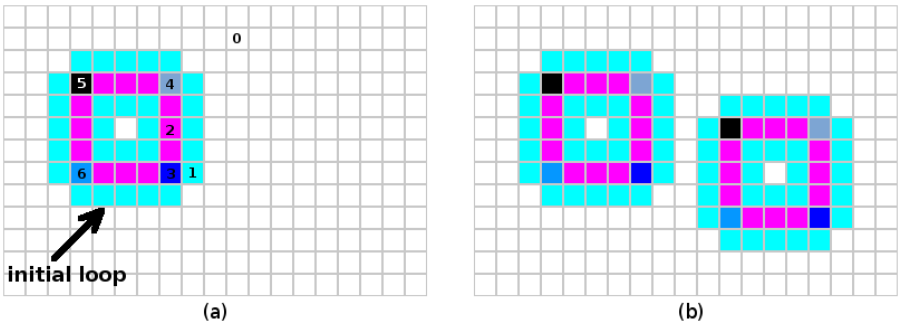


Fig. 3. The structure of the big loop in the cellular automaton that was evaluated during evolution: (a) the initial CA state containing the loop to be replicated, (b) the target state specifying the replica arrangement.

Table 1. Results of the evolutionary experiments considering the design of transition functions for the replication of the loop from Figure 3a. Success rate – the number of successful experiments out of 100 independent experiments performed that has met the fitness specification in a limit of 3 million generations, Replicates repeatedly – the number of results from the successful experiments that are able to produce more replicas during the subsequent CA development, Min. steps – the minimal number of steps of the CA needed to create the replica (i.e. the lowest value of this parameter from the group of "Replicates repeatedly" solutions, Min. rules – the minimal number of table-based transition rules obtained (i.e. the lowest value of this parameter from the group of "Replicates repeatedly" solutions.

	CA with 8 cell states				CA with 10 cell states			
Num. of CMRs	Success rate [%]	Replicates repeatedly	Min. steps	Min. rules	Success rate	Replicates repeatedly	Min. steps	Min. rules
20	0	-	-	-	1	0	-	-
30	10	6	19	84	12	9	21	146
40	9	4	20	139	12	6	16	186
50	10	6	18	130	12	6	21	177

in 16 steps (the best result achieved for this loop) whilst some CA require 30 steps (the maximal allowed number of steps) in order to finish the replication. Similarly, the number of transition rules generated from the CMRs varies from 84 to more than 1500 rules. These results indicate that cellular automata can in some cases exhibit behaviour that has not yet been discovered which may be beneficial not only for the area of CA but also, for the study of complex systems in general.

Figure 4 shows a CA development performed by one of the successful transition functions obtained for the replication of the given loop. It is one of the best

10

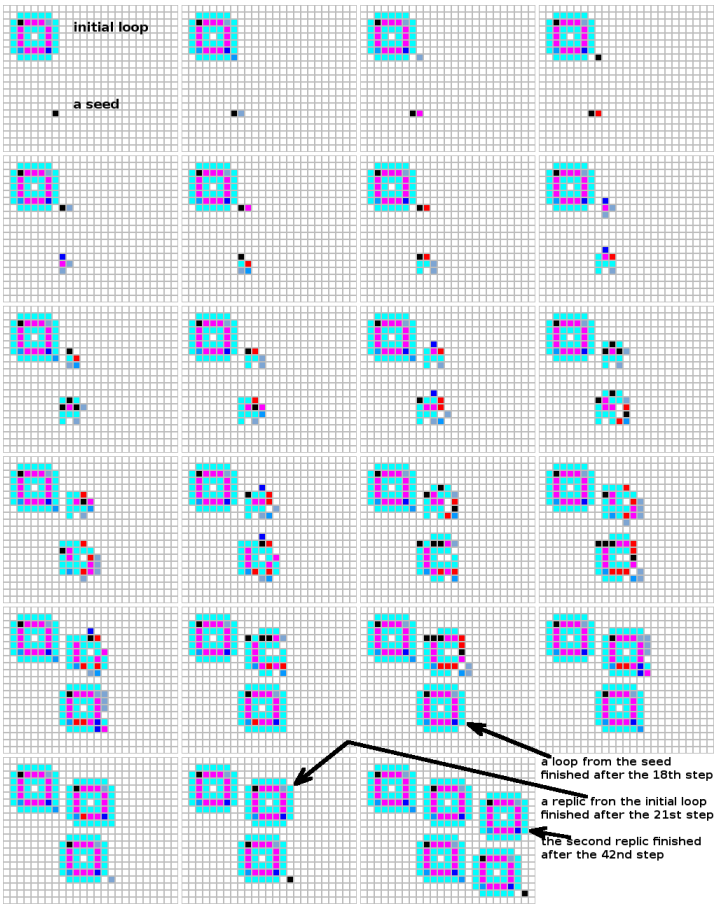


Fig. 4. Develpment of a CA performing replication of the loop from Figure 3a. The sequence of steps reads from left to right and top to bottom. The upper part of each step of the CA illustrates the replication of the initial loop. The bottom part demonstrates a seed represented by a cell in state 5. Note that after the loop is finished, its replication continues in the same way as from the initial instance (shown by the last CA state).

solutions discovered in this work with respect to the number of steps needed to create a copy of the loop. The transition function was found with 30 CMRs in the GA chromosomes and the corresponding conventional representation contains 238 transition rules. If the development of the initial loop is considered (see the upper parts of each step in Figure 4), the CA needs 21 steps to create a complete replica. As shown by the last step, more replicas can be created in the same way according to the original specification if the CA development continues. However, a more detailed investigation of this result showed that the complete initial loop is not strictly needed in order to successfully perform the

replication. For example, the loop is able to emerge even from a single seed – the lower parts of each step presented in Figure 4 shows a development of the loop from a single initial cell (a seed) in state 5. As marked by the up-most black arrow a complete loop is developed from the seed after 18 steps which is by 3 steps faster compared to the development from the initial loop. This behaviour is caused by a need of the initial loop to generate a cell in state 5 (i.e. the same state as the seed) from which the replica can be developed (it takes 3 steps – see the top-right CA state in Figure 4). The process of finishing the replica is identical with the development from the seed. Note that the ability of the transition function to develop and replicate the loop from a seed was not explicitly required in the fitness evaluation. Hence it can be considered as an additional, unconventional feature of this solution.

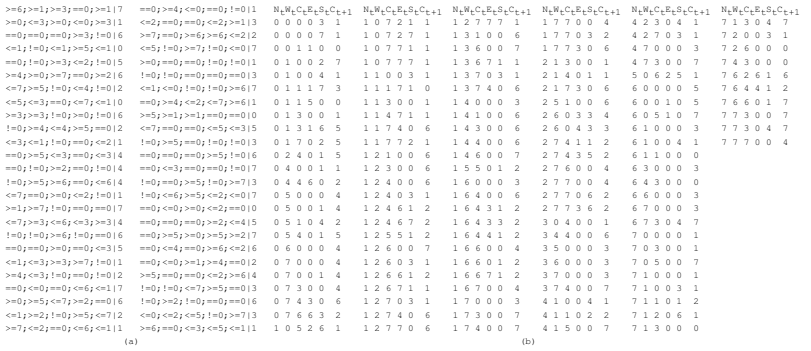


Fig. 5. Transition function for the CA in Fig. 6 and 7: (a) the evolved representation with 50 CMRs, (b) the corresponding conventional representation consisting of 130 rules. This result represents one of the best solutions discovered for the replication of the big loop.

Another result is presented in the form of an evolved transition function (Fig. 5) and the appropriate CA development (Figures 6 and 7). This cellular automaton demonstrates a development process from a seed that at first creates rather a chaotic structure even larger than the required loop itself. A "mature" loop is developed from this structure during the subsequent CA development that is able to replicate itself. Whilst the replication of the initial loop takes 25 steps (marked by the black arrow in Figure 6), the development of the chaotic structure needs 36 steps. Starting by step 37 (Fig. 7) the loop is developed from that structure in the same way as from the initial loop. It was verified that the loops are able to replicate repeatedly if the CA development continues.

For both the presented solutions the transition function was identified as redundant (i.e. not all the conventional transition rules generated from the CMR representation are needed for the replication of the initial loop required by the fitness function). A more detailed analysis showed that this redundancy is caused

12

Fig. 6. Part 1 of the replication according to the transition function from Figure 5. The sequence of steps reads from left to right and top to bottom. The development shows a replication of the initial loop (the upper part of each step) and a growth of a non-specific structure from a seed allowing to create the loop autonomously (the lower part of each step). The seed is represented by a cell in state 7.

by the finite CA size with cyclic boundary conditions and by generating the transition rules from the CMRs until the CA reaches a stable or periodic state. Although this approach leads to more complex table-based transition functions, in this case it showed as very beneficial for achieving some additional features that were not required during evolution (especially the ability to develop the loops from a seed). Advanced experiments with the resulting CA showed that if the transition functions are optimized (i.e. only the rules for the development of a single replica from the initial loop are considered), the CA in most cases loose the ability of the development from the seed. It was also determined that the seed-

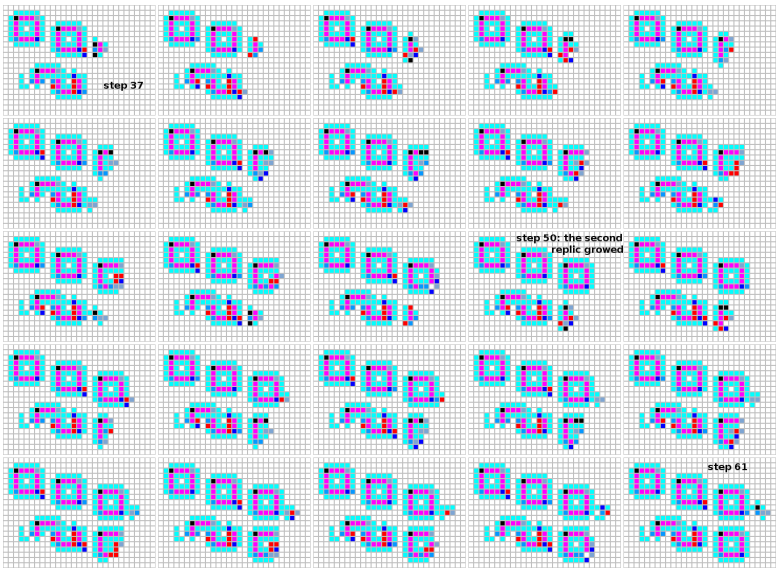


Fig. 7. Part 2 of the replication according to the transition function from Figure 5. The sequence of steps reads from left to right and top to bottom. The development shows an autonomous growth of the loop from a non-specific structure that emerged in the last step of Figure 6 (the bottom part of each step). It was verified that the loop is able to replicate in the same way as the initial loop during the subsequent CA development.

based development does not work in case of the known replicating loops (e.g. Langton's or Byl's loop). In the future, this ability may be beneficial for the advanced study of complex systems in which a given (complex) configuration needs to be achieved — distributed — from a single cell or a simple initial configuration. In addition to the results presented herein, various other solutions were found that are able to replicate a given structure. It indicates that the replication in CA is not limited to known schemes only but can be performed in many different ways.

5.2 Parallel Replication of the Small Loop

The second set of experiments presents the evolution of parallel replication techniques of a small loop with its structure shown in Figure 8a. As with the evolution of the big loop, the CA behaviour is evaluated for 30 steps using the partial fitness calculated after each step with respect to the target arrangement of the replicas shown in Figure 8b, and the final fitness value is given by the maximum of the partial fitness values. In this case, however, two replicas are required with the arrangement on the left and right side of the original loop. The hypothesis evaluated herein is that if suitable transition functions exist for the development

of the replicas, then at least a subset of the results will produce the replicas repeatedly in the given directions during the subsequent CA development (i.e. for the purposes of this study, such the solutions will be considered as general). Since the loop is not fully symmetric with respect to the cell states on the sides of the loop, it is expected that different replication algorithms (i.e. sequences of the CA steps) need to be designed to produce the replicas.

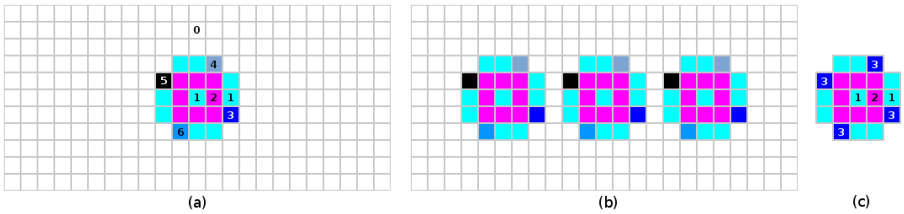


Fig. 8. The structure of the small loop in the cellular automaton that was evaluated during evolution: (a) the initial CA state containing the loop to be replicated, (b) the target state specifying the replicas arrangement, (c) example of a symmetric loop.

Table 2 summarises the results of experiments with the small loop and provides an overview of some basic parameters of the CA that can be observed during its development using the evolved transition functions. Although the shape of the small loop is simpler than the big loop, the requirement of two independent replicas increases the overall complexity of this task, the maximum success rate achieved does not exceed 12%. Despite this fact, the evolution provided some solutions that perfectly fulfil the target specification and, in addition, also exhibit the capability of the seed-based development which was not explicitly required.

Table 2. Results of the evolutionary experiments considering the design of transition functions for the replication of the loop from Figure 8a. The success rate, the number of general solutions, the minimal number of transition rules and the minimal number of CA steps needed to create the replicas were evaluated.

	CA with 8 cell states				CA with 10 cell states			
Num. of CMRs	Success rate [%]	Replicates repeatedly	Min. steps	Min. rules	Success rate	Replicates repeatedly	Min. steps	Min. rules
20	0	-	-	-	2	0	-	-
30	9	5	18	134	9	3	17	120
40	7	6	17	123	9	4	17	134
50	8	4	18	157	12	7	17	219

Figure 9 shows a CA that performs a successful parallel replication of the small loop. The CA works with 8 cell states and, in addition to the replication of the initial loop, is also able to perform the development and replication of the loop from a seed. This is one of the most efficient and compact solution obtained in this study regarding the number of CA steps and the number of transition rules. The corresponding table-based transition function consists of 154 rules as shown in Figure 10. The CA needs to perform 23 steps in order to finish the replicas of the initial loop. However, if a cell is initialised as a seed by one of the states 1, 3, 5, 6, or 7, the small loop autonomously grows into its full shape and subsequently is able to replicate according to the original specification. The analysis of the seed-based development showed that the loop needs 19 steps to fully develop from state 1, 18 steps from states 3, 6 and 7, and 24 steps from state 5. An interesting behaviour of the CA can be observed after finishing the seed development when the loop ought to be replicated. In particular, the loop replicates according to the given specification from states 1, 3, 6, and 7. However, the state-5 seed creates an undesirable structure that prevents the loop replication to the left side, i.e. the loop developed from state 5 can replicate to the right side only (see Figure 11). This indicates that a wide range of states used as the seed allows emerging the loop using various processes (i.e. sequences of CA states), which are totally different from the processes of replication from the complete loop. Although the state-5 seed does not enable to replicate the loop to both sides, the solution can be considered as robust because the undesirable structure does not cause the destruction of the loop that can subsequently replicate to the right side.

As an example of our research regarding the optimisation of replication techniques in cellular automata, a symmetric loop is considered as shown in Figure 8c. Although the evaluation method applied to design the CA for this loop is out of the scope of this study, a result of a successful parallel replication will be presented, which demonstrates the potential of the GA in combination with the CMR encoding to discover novel techniques in cellular automata. As in the previous example, the goal of the experiment was to design transition rules for the parallel replication of the loop to the given directions. Since the loop is symmetric with respect to the arrangement of the cell states, it would be possible to adapt a single replication algorithm to perform the replication process simultaneously to various directions. Such adaptation is based on "rotating" the transition rules according to the ordering of cells in the cellular neighbourhoods with respect to the given directions as known from Byl's loop [Byl, 1989]. However, if the evaluation of the candidate solutions during evolution is performed with respect to the number and arrangement of the replicas only, then the GA can discover various independent replication algorithms as shown in Figure 12. The corresponding transition function contains 137 table-based rules and is shown in Figure 13. In this solution, not only the algorithms for the replication to the left and right side differ significantly, the number of steps needed to create the replica on the left side is nearly the double of the number of steps required for the replication to the right side. As evident from Figure 12, the first replica of the initial loop

is created on the right side after the 15th step, the first replica on the left side needs 26 steps to be completed. After the 27th step, the second replica on the right side is completed whilst the second copy on the left side has just started to develop. Such a process has never been observed before as regards the known replicating loops and hence it can be considered as an unconventional feature of the solution obtained in this experiment.

5.3 Summary and Discussion

Both the proposed loops proved the ability to replicate according to the given specification. It is worth to note that although the development of the loop from a seed was not explicitly required, the evaluation of the results obtained for both the loops showed that this ability is not rare. This means that the seed-based development may be evolved directly (without any initial loop available) in order the given loop can emerge autonomously. Some experiments were performed in order to validate this hypothesis, with the following observations. The GA is able to discover transition rules for the development of the given loop from the seed. However, no solution has yet been achieved that would be able to subsequently replicate the loop. One of the reasons for this issue may be the fact that the exact place in the cellular space, where the loop is developed from the seed, is hard to predict (it depends on the state of the seed, shape of the loop and the transition rules). Therefore, it is not evident how the replicas ought to be specified within the target CA state for the continuous replication. More research is needed in order to determine the necessary information provided to the GA, which would enable to solve this problem.

In order to perform a general evaluation of the results obtained within the context of computational features of cellular automata and with respect to the existing replicating loops, the following issues need to be clarified:

1. The objective was not to design self-replication. The loops with the ability to self-replicate contain the information of how to create a copy encoded in their "body" as a suitable arrangement of cell states. The transition rules interpret this information and calculate the appropriate state transitions of the CA in order to perform the replication process. In this work, however, the initial loop is considered as an object of a given shape that ought to be transformed onto a CA state that contains the copy of the loop. The goal was to find both the transition rules and the sequence of the CA states that lead to the emergence of the replica.
2. The resulting CA do not represent universal computing models (it was not a goal of the experiments). It means that a specific transition function, that was obtained as a result of a successful evolution, is dedicated to replicate the given loop only that was a subject of evaluation in the fitness function. Nevertheless, as the results showed, some transition functions are able to create the loops from a seed which was not explicitly required within the fitness evaluation.

Although the shape of the proposed loops was inspired by the existing (self-replicating) loops and the GA provided some successful results to replicate the loops with respect to the given specifications, no working solution has yet been achieved by the GA to replicate the existing loops (e.g. Byl's loop) with the exact shape and arrangement of the replicas. This issue can be caused by the fact that some of the self-replicating loops are dynamical structures even after the replica is finished (e.g. Byl's loop exhibits such feature). However, only static replicas were considered in our experiments. Another aspect may be the size of the loop. Large loops require a considerable number of steps to finish the replica (e.g. Langton's loop needs 151 steps), which makes the evaluation of such solutions very time-consuming. Finally, the information encoded in the loop body, that specifies the self-replication features, actually determines the replication algorithm (i.e. the CA development) which is specific for the given loop. If no more valid replication algorithms exist in the solution space for a given loop, then the GA may not be able to find the solution in a reasonable time.

6 Conclusions

In summary, the results presented in this work shows several facts related to the problem of replication in cellular automata. First, there are plenty of transition functions that are able to replicate a given loop. The experiments showed that it is possible to discover such functions routinely by means of the genetic algorithm even for complex multi-state cellular automata (herein demonstrated for CA working with 8 and 10 cell states). This was enabled by the utilisation of conditionally matching rules as a technique for the representation of the transition functions. Second, some unconventional features of the solutions were identified that cannot be observed in the known replicating loops and have never been published before. Specifically, in case of some solutions obtained, the CA can be initialised by a single-cell seed in a non-zero state, which allows developing the given loop that is subsequently able to replicate. Note that this ability was identified as an extra feature of the resulting cellular automata, which was not explicitly required by the specification for the evolutionary algorithm. This shows that some cellular automata are able, using a minimum information encoded in the initial state, to autonomously develop a complex emergent behaviour that is fully determined by the transition function and the state of a single cell only. Another feature, that was achieved by the evolution, is a parallel replication of the given loop into more directions, using different algorithms to create the replicas. The results showed that this behaviour is needed if the arrangement of the cell states in the loop is not fully symmetrical. However, an unconventional parallel replication can be observed even in case of a symmetric loop, where the difference is both in the way of the replication and the number of steps needed to create the replicas. Again, the evolution itself discovered such the behaviour just on the basis of the given target pattern containing the replicas of the initial loop.

The results obtained bring some open questions, the answers of which could be beneficial for the research of cellular automata in general. For example, can the seed-based development create a configuration in the CA that supports self-replication (or other useful features)? Are there other (simple) structures that support development of more complex (self-)replicating objects? Can evolutionary techniques be applied to the design of computationally universal CA-based models? Not only these questions represent ideas for our future work.

Acknowledgements

This work was supported from IT4Innovations excellence in science project (IT4I XS LQ1602).

References

[Berlekamp et al., 2004] Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning Ways for Your Mathematical Plays, 2nd Ed., Volume 4*. A K Peters/CRC Press.

[Bidlo, 2014] Bidlo, M. (2014). Evolving multiplication as emergent behavior in cellular automata using conditionally matching rules. In *2014 IEEE Congress on Evolutionary Computation*, pages 2001–2008. IEEE Computational Intelligence Society.

[Bidlo and Vasicek, 2013] Bidlo, M. and Vasicek, Z. (2013). Evolution of cellular automata with conditionally matching rules. In *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*, pages 1178–1185. IEEE Computer Society.

[Byl, 1989] Byl, J. (1989). Self-reproduction in small cellular automata. *Physica D: Nonlinear Phenomena*, 34(1–2):295–299.

[Elmenreich and Fehérvári, 2011] Elmenreich, W. and Fehérvári, I. (2011). Evolving self-organizing cellular automata based on neural network genotypes. In *Proceedings of the 5th International Conference on Self-organizing Systems*, pages 16–25. Springer.

[Fogel et al., 1966] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. Wiley, New York.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[Langton, 1984] Langton, C. G. (1984). Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1–2):135–144.

[Mitchell et al., 1993] Mitchell, M., Hraber, P. T., and Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7(2):89–130.

[Packard, 1988] Packard, N. H. (1988). Adaptation toward the edge of chaos. In *J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, Dynamic Patterns in Complex Systems*, pages 293–301. World Scientific.

[Perrier et al., 1996] Perrier, J.-Y., Sipper, M., and Zahnd, J. (1996). Toward a viable, self-reproducing universal computer. *Physica D*, 97:335–352.

[Reggia et al., 1993] Reggia, J. A., Armentrout, S. L., Chou, H.-H., and Peng, Y. (1993). Simple systems that exhibit self-directed replication. *Science*, 259(5099):1282–1287.

[Sapin et al., 2010] Sapin, E., Adamatzky, A., Collet, P., and Bull, L. (2010). Stochastic automated search methods in cellular automata: the discovery of tens of thousands of glider guns. *Natural Computing*, 9(3):513–543.

[Sapin and Bull, 2008] Sapin, E. and Bull, L. (2008). Searching for glider guns in cellular automata: Exploring evolutionary and other techniques. In Monmarch, N., Talbi, E.-G., Collet, P., Schoenauer, M., and Lutton, E., editors, *Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, pages 255–265. Springer Berlin Heidelberg.

[Sipper, 1995] Sipper, M. (1995). Quasi-uniform computation-universal cellular cutomata. In *Advances in Artificial Life, ECAL 1995, Lecture Notes in Computer Science, Vol. 929*, pages 544–554. Springer Berlin Heidelberg.

[Sipper, 1997] Sipper, M. (1997). *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, Vol. 1194*. Springer, Berlin.

[Sipper et al., 1997] Sipper, M., Goeke, M., Mange, D., Stauffer, A., Sanchez, E., and Tomassini, M. (1997). The firefly machine: online evolware. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 181–186.

[Stefano and Navarra, 2012] Stefano, G. D. and Navarra, A. (2012). Scintillae: How to approach computing systems by means of cellular automata. In *Cellular Automata for Research and Industry*, Lecture Notes in Computer Science, Vol. 7495, pages 534–543. Springer.

[Tempesti, 1995] Tempesti, G. (1995). A new self-reproducing cellular automaton capable of construction and computation. In *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, Vol. 929, pages 555–563. Springer.

[von Neumann, 1966] von Neumann, J. (1966). *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press.

[Yunès, 2010] Yunès, J.-B. (2010). Achieving universal computations on one-dimensional cellular automata. In *Cellular Automata for Research and Industry*, Lecture Notes in Computer Science Volume 6350, pages 660–669. Springer.

20

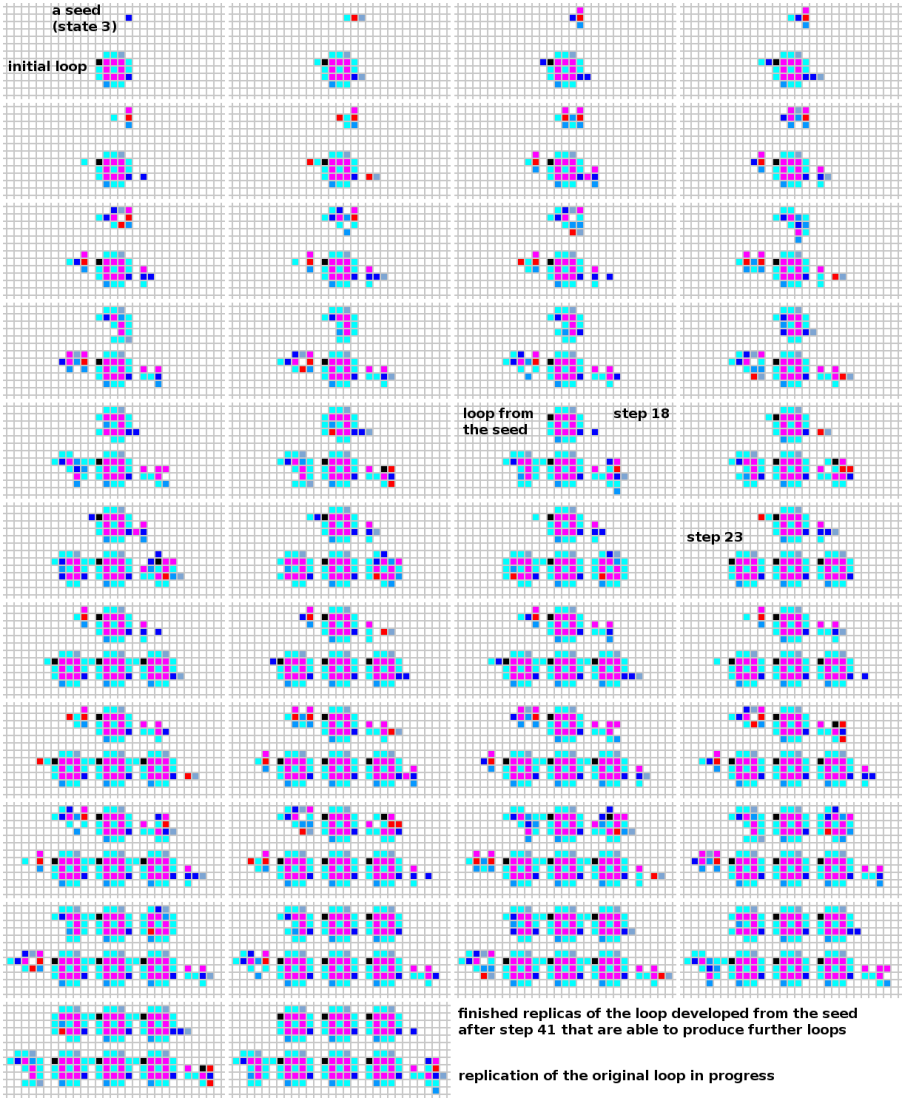


Fig. 9. A sequence of CA steps demonstrating the parallel replication of the small loop according to the evolved transition function from Figure 10. The states are ordered from left to right and top to bottom. The bottom part of each state shows the replication from the initial loop, the top part of each state demostrates the development and replication of the loop from a seed.

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	0	0	1	0	7
0	0	0	4	0	3
0	0	0	7	0	3
1	0	0	1	0	7
2	0	0	1	0	7
3	0	0	1	0	7
5	0	0	0	0	7
5	0	0	1	0	7
6	0	0	0	0	7
0	0	1	1	0	0
0	0	2	1	0	0
0	0	2	3	0	0
0	0	3	1	0	0
0	0	3	4	0	0
0	0	3	7	0	0
4	0	3	0	0	1
0	0	6	1	0	0
0	0	7	1	0	0
1	0	7	0	0	1
1	0	7	1	0	3
2	0	7	1	0	3
2	0	7	7	0	3
3	0	7	1	0	3

N_t	W_t	C_t	E_t	S_t	C_{t+1}
4	0	7	0	0	1
6	0	7	0	0	1
0	0	0	4	1	3
0	0	1	7	1	3
1	0	3	0	1	1
0	0	3	0	3	4
0	0	4	4	3	1
0	0	4	7	3	1
1	0	4	0	3	1
0	0	0	0	4	1
2	0	2	3	4	0
0	0	0	0	7	1
0	0	0	1	7	2
0	0	0	3	7	2
0	0	1	0	7	3
1	0	1	0	7	3
0	0	7	1	7	0
2	0	7	1	7	0
2	1	0	1	0	4
3	1	0	7	0	4
5	1	0	1	0	5
5	1	0	7	0	5
0	1	1	0	0	7

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	1	1	1	0	0
0	1	3	0	0	7
0	1	4	1	0	0
7	1	4	4	0	5
0	1	5	1	0	0
0	1	5	3	0	0
0	1	7	1	0	0
1	1	7	0	0	1
4	1	7	0	0	1
0	1	3	3	2	1
0	1	0	2	3	3
0	1	0	0	4	1
0	1	0	3	4	3
2	1	4	4	4	2
2	1	5	1	4	2
7	1	5	3	4	2
0	1	7	1	4	2
3	1	7	4	4	2
0	1	0	0	5	1
1	1	0	4	5	4
3	1	3	2	5	2
0	1	0	7	7	3
4	1	0	1	7	4

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	1	1	0	7	3
1	1	4	1	7	2
5	1	4	1	7	2
2	1	5	2	7	2
1	2	3	4	1	2
1	2	4	5	1	2
2	2	4	1	1	2
2	2	4	3	1	2
1	2	3	4	4	2
1	2	7	1	5	2
1	2	1	0	7	3
1	2	4	0	7	3
0	3	0	0	0	7
2	3	0	1	0	4
2	3	0	7	0	4
3	3	0	6	0	4
4	3	0	1	0	4
4	3	0	7	0	4
7	3	0	6	0	5
0	3	4	1	0	0
7	3	4	0	0	1
0	3	7	1	0	0
0	3	7	4	0	0

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	3	7	7	0	0
1	3	4	4	2	2
1	3	7	1	2	2
2	3	4	2	3	2
0	3	0	3	4	3
2	3	0	1	6	4
0	3	2	4	6	1
1	3	0	7	7	7
2	3	0	1	7	4
2	3	1	2	7	4
0	4	0	0	0	5
0	4	4	1	0	0
0	4	5	0	0	7
1	4	7	7	0	3
0	4	0	2	1	4
0	4	0	7	1	4
0	4	4	2	1	2
1	4	4	0	1	1
1	4	4	0	3	1
4	4	0	1	7	4
1	4	1	0	7	4
2	4	1	3	7	4
1	4	3	0	7	4

N_t	W_t	C_t	E_t	S_t	C_{t+1}
4	5	3	7	0	1
0	5	0	0	1	1
1	5	4	1	1	2
1	5	3	0	3	1
0	5	1	0	7	4
2	5	6	1	7	4
4	6	4	3	0	1
0	7	0	0	0	1
0	7	0	1	0	1
0	7	0	7	0	1
1	7	0	0	0	7
0	7	1	0	0	7
0	7	1	1	0	7
1	7	1	0	0	7
7	7	1	4	0	6
0	7	4	1	0	1
4	7	7	1	0	1
7	7	7	1	0	6
0	7	1	2	1	7
0	7	1	0	4	4
2	7	2	1	4	4
0	7	7	4	6	2

Fig. 10. A transition function designed by evolution for the parallel replication of the small loop from Figure 8a.

Fig. 11. A sample of the CA development from the seed according to the transition function from Figure 10: (a) the initial seed, (b) the small loop is developed from the seed after step 24, leaving an undesirable structure on its left side, (c) the loop creates its first replica after step 47, the undesirable structure prevents from the replication on the left side, (d) the replication to the right in progress after step 51, the structure on the left no longer changes.

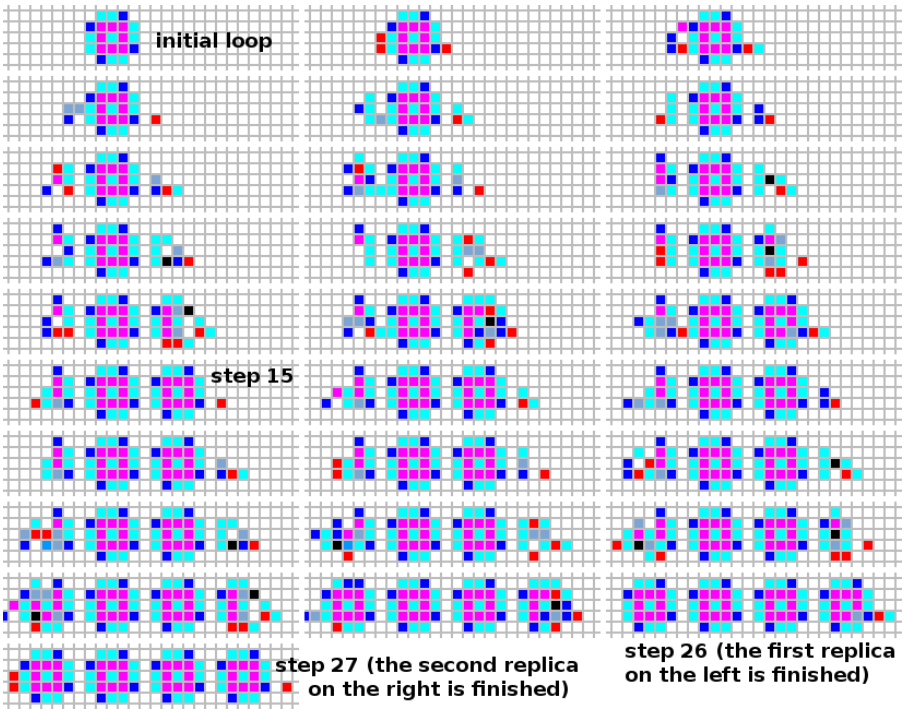


Fig. 12. A sample of the parallel replication of the symmetric loop from Figure 8c according to the transition function shown in Figure 13. Note that the number of steps needed to develop a replica on the right side is half the number of steps required to finish a replica on the left side.

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	0	0	1	0	7
0	0	0	4	0	3
0	0	0	7	0	3
1	0	0	1	0	7
2	0	0	1	0	7
3	0	0	1	0	7
5	0	0	0	0	7
5	0	0	1	0	7
6	0	0	0	0	7
0	0	1	1	0	0
0	0	2	1	0	0
0	0	2	3	0	0
0	0	3	1	0	0
0	0	3	4	0	0
0	0	3	7	0	0
4	0	3	0	0	1
0	0	6	1	0	0
0	0	7	1	0	0
1	0	7	0	0	1
1	0	7	1	0	3
2	0	7	1	0	3
2	0	7	7	0	3
3	0	7	1	0	3

N_t	W_t	C_t	E_t	S_t	C_{t+1}
4	0	7	0	0	1
6	0	7	0	0	1
0	0	0	4	1	3
0	0	1	7	1	3
1	0	3	0	1	1
0	0	3	0	3	4
0	0	4	4	3	1
0	0	4	7	3	1
1	0	4	0	3	1
0	0	0	0	4	1
2	0	2	3	4	0
0	0	0	0	7	1
0	0	0	1	7	2
0	0	0	3	7	2
0	0	1	0	7	3
1	0	1	0	7	3
0	0	7	1	7	0
2	0	7	1	7	0
2	1	0	1	0	4
3	1	0	7	0	4
5	1	0	1	0	5
5	1	0	7	0	5
0	1	1	0	0	7

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	1	1	1	0	0
0	1	3	0	0	7
0	1	4	1	0	0
7	1	4	4	0	5
0	1	5	1	0	0
0	1	5	3	0	0
0	1	7	1	0	0
1	1	7	0	0	1
4	1	7	0	0	1
0	1	3	3	2	1
0	1	0	2	3	3
0	1	0	0	4	1
0	1	0	3	4	3
2	1	4	4	4	2
2	1	5	1	4	2
7	1	5	3	4	2
0	1	7	1	4	2
3	1	7	4	4	2
0	1	0	0	5	1
1	1	0	4	5	4
3	1	3	2	5	2
0	1	0	7	7	3
4	1	0	1	7	4

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	1	1	0	7	3
1	1	4	1	7	2
5	1	4	1	7	2
2	1	5	2	7	2
1	2	3	4	1	2
1	2	4	5	1	2
2	2	4	1	1	2
2	2	4	3	1	2
1	2	3	4	4	2
1	2	7	1	5	2
1	2	1	0	7	3
1	2	4	0	7	3
0	3	0	0	0	7
2	3	0	1	0	4
2	3	0	7	0	4
3	3	0	6	0	4
4	3	0	1	0	4
4	3	0	7	0	4
7	3	0	6	0	5
0	3	4	1	0	0
7	3	4	0	0	1
0	3	7	1	0	0
0	3	7	4	0	0

N_t	W_t	C_t	E_t	S_t	C_{t+1}
0	3	7	7	0	0
1	3	4	4	2	2
1	3	7	1	2	2
2	3	4	2	3	2
0	3	0	3	4	3
2	3	0	1	6	4
0	3	2	4	6	1
1	3	0	7	7	7
2	3	0	1	7	4
2	3	1	2	7	4
0	4	0	0	0	5
0	4	4	1	0	0
0	4	5	0	0	7
1	4	7	7	0	3
0	4	0	2	1	4
0	4	0	7	1	4
0	4	4	2	1	2
1	4	4	0	1	1
1	4	4	0	3	1
4	4	0	1	7	4
1	4	1	0	7	4
2	4	1	3	7	4
1	4	3	0	7	4

N_t	W_t	C_t	E_t	S_t	C_{t+1}
4	5	3	7	0	1
0	5	0	0	1	1
1	5	4	1	1	2
1	5	3	0	3	1
0	5	1	0	7	4
2	5	6	1	7	4
4	6	4	3	0	1
0	7	0	0	0	1
0	7	0	1	0	1
0	7	0	7	0	1
1	7	0	0	0	7
0	7	1	0	0	7
0	7	1	1	0	7
1	7	1	0	0	7
7	7	1	4	0	6
0	7	4	1	0	1
4	7	7	1	0	1
7	7	7	1	0	6
0	7	1	2	1	7
0	7	1	0	4	4
2	7	2	1	4	4
0	7	7	4	6	2

Fig. 13. The transition function designed by evolution for the parallel replication of the symmetric loop from Figure 8c.

Appendix VI

Advances in the Evolution of Complex Cellular Automata

BIDLO Michal

In: *International Joint Conference on Computational Intelligence, IJCCI 2016, Porto, Portugal, November 9-11, 2016 Revised Selected Papers.* Cham: Springer International Publishing, 2017, pp. 123-146. ISBN 978-3-319-99282-2.

This work represents an extended version of [20] which was **awarded by the Best Paper Award** in 2016 International Joint Conference on Computational Intelligence (IJCCI), Porto, Portugal, and selected for publication of an extended version hereafter.

Advances in the Evolution of Complex Cellular Automata

Michal Bidlo

Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, 61266 Brno, Czech Republic
E-mail: bidlom@fit.vutbr.cz
http://www.fit.vutbr.cz/~bidlom

Abstract. In this study we present some advanced experiments dealing with the evolutionary design of multi-state uniform cellular automata. The generic square calculation problem in one-dimensional automata will be treated as one of the case studies. An analysis of the evolutionary experiments will be proposed and properties of the resulting cellular automata will be discussed. It will be demonstrated that various approaches to the square calculations in cellular automata exist, some of which substantially overcome the known solution. The second case study deals with a non-trivial pattern development problem in two-dimensional automata. Some of the results will be presented which indicate that an exact behaviour can be automatically designed even for cellular automata working with more than ten cell states. A discussion for both case studies is included and potential areas of further research are highlighted.

Keywords: evolutionary algorithm, cellular automaton, transition function, conditional rule, square calculation, pattern development

1 Introduction

The concept of cellular automata was introduced by von Neumann in [15]. One of the aspects widely studied in his work was the problem of (universal) computational machines and the question about their ability to make copies of themselves (i.e. to self-reproduce). Von Neumann proposed a model with 29 cell states to perform this task. Later Codd proposed another approach and showed that the problem of computation and construction can be performed by means of a simplified model working with 8 states only [7].

Several other researchers studied cellular automata usually by means of various rigorous techniques. For instance, Sipper studied computational properties of *binary* cellular automata (i.e. those working with 2 cell states only) and proposed a concept of universal computing platform using a two-dimensional (2D) CA with non-uniform transition function (i.e. each cell can, in general, be controlled by a different set of transition rules) [21]. Sipper showed that, by introducing the non-uniform concept to the binary CAs, universal computation can be realised, which

was not possible using the Codd's model. In fact, Sipper's work significantly reduced the complexity of the CA in comparison with the models published earlier. Nevertheless even the binary uniform 2D CAs can be computationally universal if 9-cell neighbourhood is considered. Such CA was implemented using the famous rules of the Game of Life [2] (original proof of the concept was published in 1982 and several times revisited – e.g. see [8][11][17][18]).

Although binary CAs may be advantageous due to simple elementary rules and hardware implementations in particular, many operations and real-world problems can effectively be solved by *multi-state* cellular automata (i.e. those working with more than 2 cell states) rather than those using just two states. For example, a technique for the construction of computing systems in a 2D CA was demonstrated in [23] using rules of a simple game called Scintillae working with 6 cell states. Computational universality was also studied with respect to one-dimensional (1D) CA, e.g. in [13][27].

However, in some cases application specific operations (algorithms) may be more suitable than programming a universal system, allowing to better optimize various aspects of the design (e.g. resources, efficiency, data encoding etc.). For example, Tempesti [25] and Perrier et al. [12] showed that specific arrangements of cell states can encode sequences of instructions (programs) to perform a given operation. Wolfram presented various transition functions for CAs in order to compute elementary as well as advanced functions (e.g. parity, square, or prime number generation) [26]. Further problems were investigated in recent years [16][19].

In addition to the computational tasks, various other (more geneal) benchmark problems have been investigated using cellular automata, e.g. including principles of self-organization, replication or pattern formation. For example, Basanta et al. used a genetic algorithm to evolve the rules of effector automata (a generalised variant of CA) to create microstructural patterns that are similar to crystal structures known from some materials [1]. An important aspect of this work was to investigate new materials with specific properties and their simulation using computers. Suzudo proposed an approach to the evolutionary design of 2D asynchronous CA for a specific formation of patterns in groups in order to better understand of the pattern-forming processes known from nature [24]. Elmenreich et al. proposed an original technique for growing self-organising structures in CA whose development is controlled by neural networks according to the internal cell states [9].

The proposed work represents an extended version of our recent study published in [4], the aim of which is to present a part of our wider research in the area of cellular automata, where representation techniques and automatic (evolutionary) methods for the design of complex multi-state cellular automata are investigated. The goal of this work is to design transition functions for cellular automata using evolutionary algorithms, which satisfy the given behaviour with respect to some specific initial and target conditions. In particular, it will be shown that the evolutionary algorithm can design various transition functions for uniform 1D CAs (that have never been seen before) to perform *generic*

square calculations in the cellular space using just local interactions of cells. An additional analysis of the results demonstrates that various generic CA-based solutions of the squaring problem can be discovered, which substantially overcome the known solution regarding both the complexity of the transition functions and the number of steps (speed) of calculation. In order to show the abilities of the proposed method for designing CA using the concept of conditionally matching rules, some further experiments are presented regarding the evolution of 2D multi-state cellular automata in which the formation of some non-trivial patterns is treated as a case study. As cellular automata represent a platform potentially important for future technologies (see their utilisation in various emerging fields, e.g. [14], [22] or [20]), it is worth studying their design and behaviour on the elementary level as well (i.e. using various benchmark problems).

2 Cellular Automata for Square Calculations

For the purposes of developing algorithms for squaring natural numbers, 1D uniform cellular automata are treated with the following specification (target behaviour). The number of cell states is investigated for values 4, 6, 8 and 10 (this was chosen on the basis of the existing solution [26] that uses 8 states; moreover it is worth of determining whether less states will enable to design generic solutions and whether the EA will be able to find solutions in a huge search space induced by 10 cell states). The new state of a given cell depends on the states of its west neighbour (c_W), the cell itself (central cell, c_C) and its east neighbour (c_E), i.e. it is a case of 3-cell neighbourhood. *A step* of the CA will be considered as a synchronous update of state values of all its cells according to a given transition function. For the practical implementation purposes, cyclic boundary conditions are considered. However, it is important to note that CAs with sufficient sizes are used in order to avoid affecting the development by the finite number of cells.

The value of x is encoded in the initial CA state as a continuous sequence of cells in state 1, whose length (i.e. the number of cells in state 1) corresponds to x, the other cells possess state 0. For example, the state of a 12-cell CA, which encodes $x = 3$, can appear as 0000011100000. The result $y = x^2$, that will emerge from the initial state in a finite number of steps, is assumed as a stable state in which a continuous sequence of cells in non-zero states can be detected, the length of which equals the value of y, the other cells are required in state 0. For the aforementioned example, the result can appear as 002222222220 or even 023231323200 (there is a sequence of non-zero cells of length $3^2 = 9$). The concept of representing the input value x and the result y is graphically illustrated in Figure 1. This is a generalised interpretation based on the idea presented in [26], page 639. The goal is to discover transition functions for the CA, that are able to calculate the square of arbitrary number $x > 1$.

4

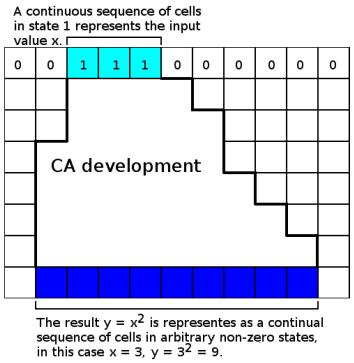


Fig. 1. Illustration of encoding integer values in a 1D cellular automaton. In this example $x = 3, y = 9$. Extracted from [4].

2.1 Conditionally Matching Rules

In order to represent the transition functions for CAs, the concept of Conditionally Matching Rules (CMR), originally introduced in [6], will be applied. This technique showed as very promising for designing complex cellular automata [3][5]. For the 1D CA working with 3-cell neighbourhood, a CMR is defined as $(cond_W\ s_W)(cond_C\ s_C)(cond_E\ s_E) \rightarrow s_{Cnew}$, where $cond_\bigstar$ denotes a condition function and $s_\bigstar$ denotes a state value. Each part $(cond_\bigstar\ s_\bigstar)$ on the left of the arrow is evaluated with respect to the state of a specific cell in the neighbourhood (in this case c_W, c_C and c_E respectively). For the experiments presented in this work the relation operators $=$, $\neq$, $\geq$ and $\leq$ are considered as the condition functions. A finite sequence of CMRs represents a transition function. In order to determine the new state of a cell, the CMRs are evaluated sequentially. If a rule is found in which all conditions are true (with respect to the states in the cell neighbourhood), s_{Cnew} from this rule is the new state of the central cell. Otherwise the cell state does not change. For example, consider a transition function that contains a CMR $(\neq 1)(\neq 2)(\leq 1) \rightarrow 1$. Let c_W, c_C, c_E be states of cells in a neighbourhood with values $2, 3, 0$ respectively, and a new state of the central cell ought to be calculated. According to the aforementioned rule, $c_W \neq s_W$ is true as $2 \neq 1$, similarly $c_C \neq s_C$ is true $(3 \neq 2)$ and $c_E \leq s_E$ $(0 \leq 1)$. Therefore, this CMR is said to match, i.e. $s_{Cnew} = 1$ on its right side will update the state of the central cell. Note that the same concept can also be applied to CA working with a wider cellular neighborhood. For example, a CMR for a 2D CA with 5-cell neighborhood would consist of 5 items for the conditional functions instead for 3 items.

The evolved CMRs can be transformed to the conventional table rules [5] without loss of functionality or violating the basic CA principles. In this work the transformation is performed as follows: (1) For every possible combination of states $c_W\ c_C\ c_E$ in cellular neighborhood a new state s_{Cnew} is calculated using the CMR-based transition function. (2) If $c_C \neq s_{Cnew}$ (i.e. the cell state ought

to be modified), then a table rule of the form $c_W\ c_C\ c_E \rightarrow s_{Cnew}$ is generated. Note that the combinations of states not included amongst the table rules do not change the state of the central cell, which is treated implicitly during the CA simulation. The number of such *generated rules* will represent a metrics indicating the complexity of the transition function.

In order to determine the complexity of the transition function with respect to a specific square calculation in CA, a set of *used rules* is created using the aforementioned principle whereas the combinations of states $c_W\ c_C\ c_E$ are considered just occurring during the given square calculation in the CA. There metrics (together with the number of states and CA steps) will allow us to compare the solutions obtained by the evolution and to identify the best results with respect to their complexity and efficiency.

An evolutionary algorithm will be applied to search for suitable CMR-based transition functions as described in the following section.

3 Setup of the Evolutionary System

A custom evolutionary algorithm (EA) was utilised, which is a result of our long-term experimentation in this area. Note, however, that neither tuning of the EA nor in-depth analysis of the evolutionary process is a subject of this work. The EA is based on a simple genetic algorithm [10] with a tournament selection of base 4 and a custom mutation operator. Crossover is not used as it has not shown any improvement in success rate or efficiency of our experiments.

The EA utilises the following fixed-length representation of the conditionally matching rules in the genomes. For the purpose of encoding the condition functions $=$, $\neq$, $\geq$ and $\leq$, integer values 0, 1, 2 and 3 will be used respectively. Each part $(cond_\bigstar\ s_\bigstar)$ of the CMR is encoded as a single integer $P_\bigstar$ in the range from 0 to M where $M = 4 * S - 1$ (4 is the fixed number of condition functions considered and S is the number of cell states) and the part $\rightarrow s_{Cnew}$ is represented by an integer in the range from 0 to $S - 1$. In order to decode a specific condition and state value, the following operations are performed: $cond_\bigstar = P_\bigstar / S$, $s_\bigstar = P_\bigstar\ mod\ S$ (note that / is the integer division and *mod* is the modulo-division). This means that a CMR $(cond_W\ s_W)(cond_C\ s_C)(cond_E\ s_E) \rightarrow s_{Cnew}$ can be represented by 4 integers; if 20 CMRs ought to be encoded in the genome, then $4 * 20 = 80$ integers are needed. For example, consider $S = 3$ for which $M = 4 * 3 - 1 = 11$. If a 4-tuple of integers (2 9 11 2) representing a CMR in the genome ought to be decoded, then the integers are processed respectively as:

- $cond_W = 2/3 = 0$ which corresponds to the operator $=$, $s_W = 2\ mod\ 3 = 2$,
- $cond_C = 9/3 = 3$ which corresponds to the operator $\leq$, $s_C = 9\ mod\ 3 = 0$,
- $cond_E = 11/3 = 3$ which corresponds to the operator $\leq$, $s_E = 11\ mod\ 3 = 2$,
- $s_{Cnew} = 2$ is directly represented by the 4th integer.

Therefore, a CMR of the form $(=\ 2)(\leq\ 0)(\leq\ 2) \rightarrow 2$ has been decoded.

The following variants of the fitness functions are treated (note that the input x is set to the middle of the cellular array):

1. RESULT ANYWHERE (RA-fitness): The fitness is calculated with respect to any valid arrangement (position) of the result sequence in the CA. For example, $y = 4$ in an 8-cell CA may be $rrrr0000$, $0rrrr000$, $00rrrr00$, $000rrrr0$ or $0000rrrr$, where $r \neq 0$ represent the result states that may be generally different within the result sequence. A partial fitness value is calculated for every possible arrangement of the result sequence as the sum of the number of cells in the expected state for the given values of x. The final fitness is the highest of the partial fitness values.
2. SYMMETRIC RESULT (SR-fitness): The result is expected symmetrically with respect to the input. For example, if 0000011100000 corresponds to initial CA state for $x = 3$, then the result $y = 3^2$ is expected as a specific CA state $00rrrrrrrr00$ (each r may be represented by any non-zero state). The fitness is the number of cells in the expected state.

The fitness evaluation of each genome is performed by simulating the CA for initial states with the values of x from 2 to 6. The result of the x^2 calculation is inspected after the 99th and 100th step of the CA, which allows to involve the state stability check into the evaluation. This approach was chosen on the basis of the maximal x evaluated during the fitness calculation and on the basis of the number of steps needed for the square calculation using the existing solution [26]. In particular, the fitness of a fully working solution evaluated for x from 2 to 6 in a 100-cell CA is given by $F_{max} = 5 * 2 * 100 = 1000$ (there are 5 different values of x for which the result x^2 is investigated in 2 successive CA states, each consisting of 100 cells). The evolved transition functions, satisfying the maximal fitness for the given range of x, are checked for the ability to work in larger CAs for up to $x = 25$ The solutions which pass this check are considered as generic.

The EA works with a population of 8 genomes initialised randomly at the beginning of evolution. After evaluating the genomes, four candidates are selected randomly, the candidate with the highest fitness becomes a parent. An offspring is created by mutating 2 randomly selected integers in the parent. The selection and mutation continue until a new population of the same size is created and the evolutionary process is repeated until 2 million generations are performed. If a solution with the maximal fitness is found, then the evolutionary run is considered as successful. If no such solution is found within the given generation limit, then the evolutionary run is terminated and regarded as unsuccessful.

4 Results of Square Calculations in 1D CA

The evolutionary design of CAs for the generic square calculation has been investigated for the following settings: the number of states 4, 6, 8 and 10, the transition functions consisting of 20, 30, 40 and 50 CMRs and two ways of the fitness calculation described in Section 3. For each setup, 100 independent evolutionary runs have been executed. The success rate and average number of generations needed to find a working solution were observed with respect to the evolutionary process. As regards the parameters of the CA, the minimal number of rules and steps needed to calculate the square of x were determined.

Table 1. Statistics of the evolutionary experiments conducted using the RA-fitness (the upper part of the table) and the parameters of the generic solutions (in the lower part of the table). The parameters of the best results obtained are marked **bold**. Note that # denotes "the number of", the meaning of "generated rules", "used rules" and "steps" of the CA is defined in Section 2. Extracted from [4].

	the number of states															
	4				6				8				10			
the num. of CMRs	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules
20	3	844364	54	35	30	769440	45	120	45	570939	39	232	35	328210	47	569
30	3	620998	52	36	24	749837	40	120	38	595467	42	340	33	363360	45	663
40	2	1344286	77	46	19	629122	37	136	30	701612	41	365	29	244566	46	662
50	2	959689	73	43	20	813803	41	134	35	582342	39	348	38	373490	40	762

	the number of generic solutions (#generic) obtained for the given number of states and parameters of the generic solutions: #generated rules/#used rules/#steps for 6^2)			
#generic,	1	5	6	3
parameters	**36/26/74**	**176/52/46**, 164/33/87, 152/49/78, 185/66/70, 175/52/69	435/49/68, 403/51/79, 422/39/65, 392/62/76, 423/41/68, 429/94/76	934/64/56, 835/61/79, 916/35/76

For the purposes of comparison of the results proposed in Section 4.3, the CA will be denominated by unique identifiers of the form CA–XX–YY, where XX and YY are integers distinguishing the sets of evolutionary experiments and the CA obtained.

4.1 Results for the RA-Fitness

For the RA-fitness, the statistical results are summarised in Table 1. The table also contains the total numbers of generic solutions discovered for the given state setups and parameters determined for these solutions. For every number of states considered, at least one generic solution was identified. For example, a transition function was discovered for the 4-state CA, which consists of 36 table rules (transformed from the CMR representation evolved). This solution can be optimised to 26 rules (by eliminating the rules not used during the square calculation) which represents the simplest CA for generic square calculations known so far (note that Wolfram's CA works with 8 states and 51 rules [26]). Moreover, for example, our solution needs 74 steps to calculate 6^2 whilst Wolfram's CA needs 112 steps, which also represents a substantial innovation discovered by the EA. The CA development corresponding to this solution is shown in Figure 2.

Another result obtained using the RA-fitness is illustrated by the CA development in Figure 3. In this case the CA works with 6 states and its transition function consists of 52 effective rules. The number of steps needed, for example, to calculate 6^2, is 46 (and compared to 112 steps of Wolframs CA, it is an improvement of the CA efficiency by more than 50%) which represents the best CA known so far for this operation and the best result obtained from our experiment.

8

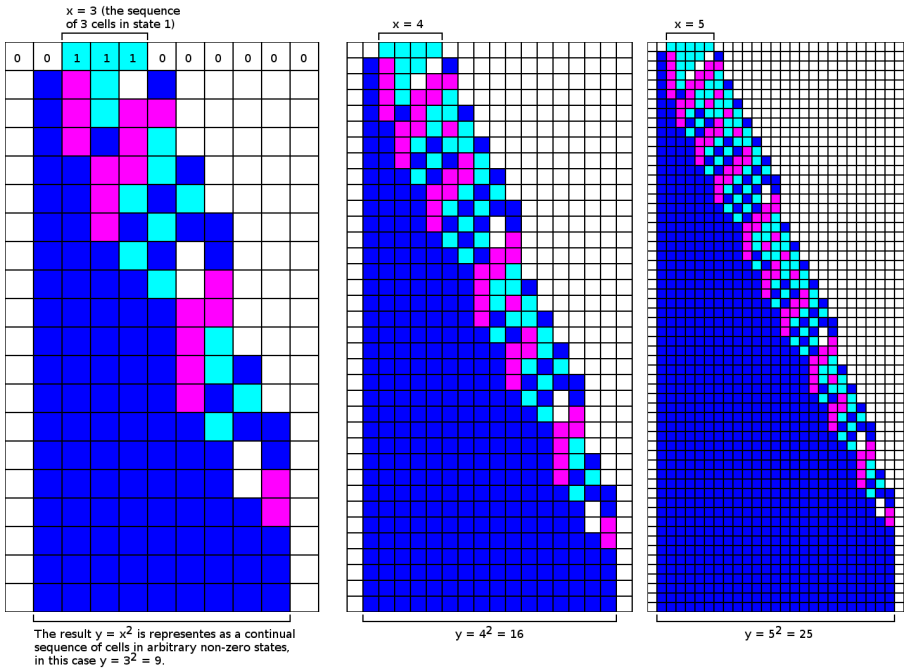


Fig. 2. Example of a 4-state squaring CA development for $x = 3, 4$ and 5 using our most compact transition function. This solution is denominated as CA–30–00 and its rules are: 0 0 1 → 3, 0 1 1 → 2, 0 3 0 → 2, 1 0 0 → 3, 1 0 2 → 2, 1 0 3 → 2, 1 1 0 → 0, 1 1 2 → 2, 1 1 3 → 2, 1 2 1 → 1, 1 3 0 → 1, 1 3 1 → 1, 1 3 2 → 2, 1 3 3 → 0, 2 1 2 → 3, 2 1 3 → 3, 2 2 0 → 1, 2 2 1 → 1, 2 3 1 → 1, 2 3 2 → 2, 3 0 2 → 3, 3 1 0 → 3, 3 1 1 → 3, 3 1 3 → 3, 3 2 0 → 3, 3 2 3 → 3. Extracted from [4].

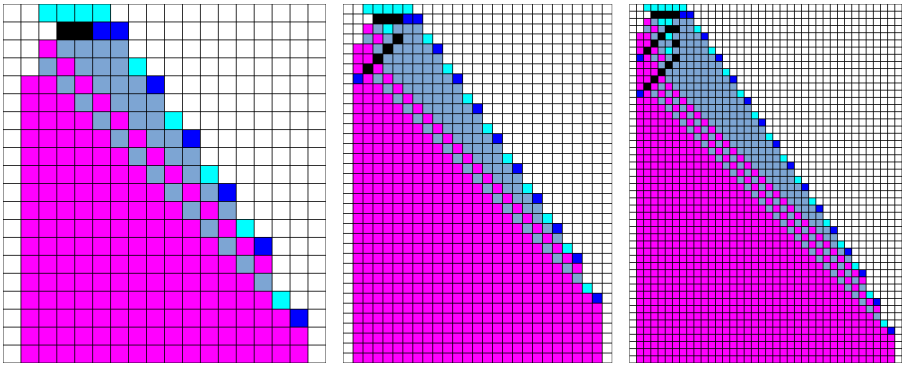


Fig. 3. Example of a 6-state CA development for $x = 4, 5$ and 6. This is the fastest CA-based (3-neighbourhood) solution known so far and the best result obtained from our experiments. This solution is denominated as CA–50–12 and its rules are: 0 0 4 → 2, 0 0 5 → 2, 0 1 1 → 0, 0 2 3 → 0, 0 2 4 → 4, 0 2 5 → 3, 0 3 2 → 2, 0 3 3 → 0, 0 4 0 → 2, 0 4 2 → 2, 0 5 3 → 0, 0 5 5 → 4, 1 0 0 → 3, 1 1 0 → 3, 1 1 1 → 5, 1 1 4 → 5, 1 4 4 → 5, 2 0 4 → 3, 2 1 0 → 2, 2 2 5 → 5, 2 3 0 → 2, 2 3 4 → 2, 2 4 0 → 2, 2 4 1 → 2, 2 4 2 → 2, 2 4 3 → 2, 2 4 4 → 2, 2 4 5 → 5, 2 5 2 → 2, 2 5 4 → 2, 3 3 0 → 4, 3 4 4 → 2, 4 0 0 → 1, 4 1 0 → 4, 4 1 1 → 4, 4 1 4 → 4, 4 2 0 → 4, 4 2 1 → 4, 4 2 3 → 4, 4 2 4 → 4, 4 3 0 → 4, 4 4 5 → 1, 4 5 1 → 4, 4 5 4 → 4, 4 5 5 → 1, 5 1 1 → 4, 5 1 4 → 4, 5 2 4 → 4, 5 3 3 → 4, 5 5 3 → 4, 5 5 4 → 4, 5 5 5 → 1. Extracted from [4].

One more example of evolved CA is shown in Figure 4. This generic solution was obtained in the setup with 8-state CA, however, the transition function works with 6 different states only. There are 49 transition rules, the CA needs 68 steps to calculate 6^2. This means that the EA discovered a simpler solution (regarding the the number of states and table rules) which is a part of the solution space of the 8-state CA. Again, this result exhibits generally better parameters compared to the known solution from [26]. The CA development, that was not observed in any other solution, is also interesting visually - as Fig. 4 shows, the CA generates a pattern with some "dead areas" (cells in state 0) within the cells that subsequently form the result sequence. The size of these areas is gradually reduced, which finally lead to derive the number of steps after which a stable state containing the correct result for the given x has emerged (illustrated by the right part of Figure 4 for $x = 8$ whereas the CA needs 122 steps to produce the result).

4.2 Results for the SR-Fitness

Table 2 shows the statistics for the SR-fitness together with the total numbers of generic solutions discovered for the given state setups and parameters determined for these solutions. As evident, the success rates are generally lower compared to the RA-fitness which is expectable because the SR-fitness allows a single arrangement only of the result sequence in the CA. Moreover, just two generic

10

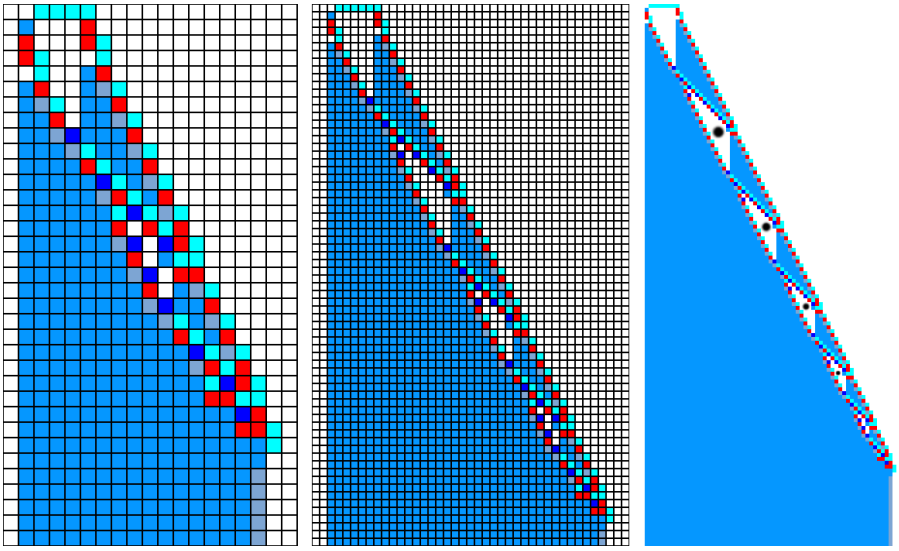


Fig. 4. Example of a 6-state squaring CA development (originally designed using 8-state setup) for $x = 4, 6$ and 8. This solution is denominated as CA–40–01. Its development shows a specific pattern evolved to derive the result of x^2, which was not observed in any other solution. The part on the right shows a complete global behaviour of this CA for $x = 8$ with some "dead areas" (marked by black spots) which lead to the correct stable result by progressively reducing the size of these areas (in this case the result of 8^2 is achieved after 122 steps). Extracted from [4].

CAs have been identified out of all the runs executed for this setup. However, the goal of this experiment was rather to determine whether solutions of this type ever exist for cellular automata and evaluate the ability of the EA to find them. As regards both generic solutions, their numbers of used rules and CA steps are significantly better in comparison with Wolfram's solution [26]. Specifically, Wolfram's solution uses are 51 rules and the calculation of 6^2 takes 112 steps, whilst the proposed results use 33, respective 36 rules and calculate 6^2 in 71, respective 78 steps. Moreover, one of them was discovered using a 4-state CA (Wolfram used 8 states), which belongs to the most compact solutions obtained herein and known so far.

Table 2. Statistics of the evolutionary experiments conducted using the SR-fitness (the upper part of the table) and the parameters of the generic solutions (in the lower part of the table). The parameters of the best result obtained are marked **bold**. Note that # denotes "the number of", the meaning of "generated rules", "used rules" and "steps" of the CAs is defined in Section 2. Extracted from [4].

	the number of states															
	4				6				8				10			
the num. of CMRs	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules	succ. rate	avg. gen.	min. steps	min. rules
20	2	634948	71	38	4	734200	38	126	11	982446	34	234	18	855791	53	542
30	0	-	-	-	5	905278	48	150	17	934123	51	327	15	910269	35	742
40	1	1546681	79	45	4	928170	33	147	11	1033059	53	317	15	898314	52	748
50	0	-	-	-	3	989039	44	138	12	811686	32	380	17	861850	52	796
	the number of generic solutions (#generic) obtained for the given number of states and parameters of the generic solutions: #generated rules/#used rules/#steps for 6^2)															
#generic, parameters	1 **38/33/71**				0				1 234/36/78				0			

Figure 5 shows examples of a CA (identified as generic) evolved using the SR-fitness. The transition function, originally obtained in 8-state CA setup, is represented by 36 used rules and works with 7 states only. Although this result cannot be considered as very efficient (for 6^2 the CA needs 78 steps), it exhibits one of the most complex emergent process obtained for the square calculation, the result of which is represented by a non-homogeneous state. The sample on the right of Fig. 5 shows a cutout of development for $x = 11$ in which the global behaviour can be observed. This result demonstrates that the EA can produce generic solutions to a non-trivial problem even for a single specific position of the result sequence required by the SR-fitness evaluation.

4.3 Analysis and Comparison of the Results

In this section an overall analysis and comparison of the results obtained for the generic square calculations is provided and some of further interesting CA are shown. Note that the data related to the CA behavior and visual samples

12

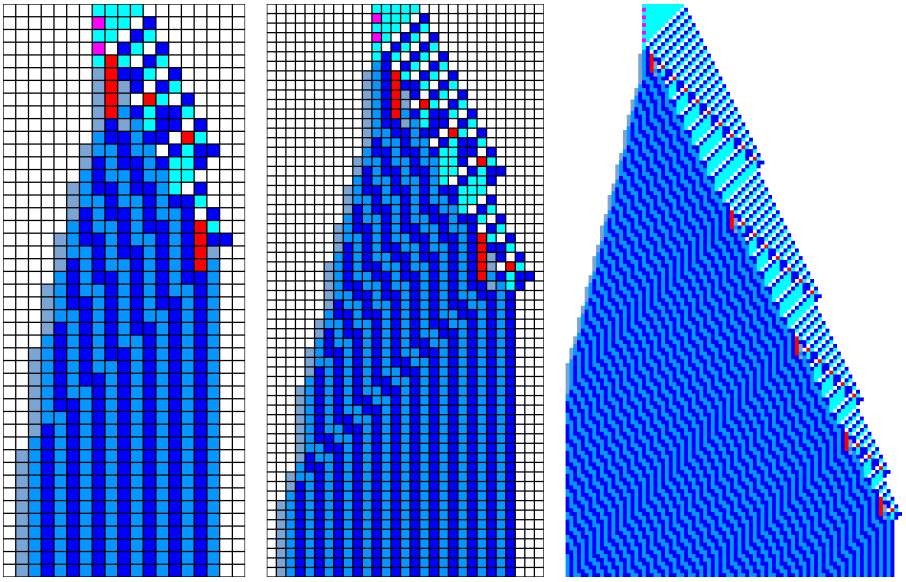


Fig. 5. Example of a 7-state CA controlled by a transition function evolved using the SR-fitness. This solution is denominated as CA–20–07. A complete development is shown for $x = 4$ and 5 (the left and middle sample respectively), the part on the right demonstrates a cutout of global behaviour of the CA for $x = 11$. Extracted from [4].

of selected calculations were obtained using our experimental software (i.e. the evolutionary system and a dedicated CA simulator developed specifically for this purpose). There are in total 17 different CA obtained from our experiments and included in this evaluation.

In order to provide a direct comparison of computational efficiency of the CA, the number of steps needed to calculate the square was evaluated for the values of the input integer x from 2 to 16. We used the WolframAlpha computational knowledge engine[1] to generate expressions which allow us to determine the number of steps of a given CA for $x > 16$. Tables 3 and 4 summarize the analysis. The resulting CA are sorted from the best to worst regarding the number of steps for $x = 16$ as the sorting criterion.

For some CA the equations derived by WolframAlpha are specified for an independent integer variable $n > 0$ by means of which the number of steps can be determined for a given input value of x. For example, the number of steps a_n of the CA–50–27 from Table 3 can be determined as $a_n = n(2n + 3)$ for all $x \geq 1$, i.e. in order to calculate the number of steps for $x = 7$, for example, then $n = x - 1 = 6$ and the substitution of this value to the equation gives $a_n = 6 * (2 * 6 + 3) = 90$ steps which corresponds to the value from Table 3 for $x = 7$ (observed for this CA using our CA simulator). The reason for taking $n = x - 1$ follows from the fact that the cellular automata work for $x \geq 2$.

For some CA WolframAlpha derived an iterative expression determining the number of steps a_{n+1} from the previous value a_n. For example, the CA–50–12 (the best one in this work) allows determining the number of steps for a given x as $a_{n+1} = n(3n + 7) - a_n$ for $n \geq 2$. Therefore, in order to calculate the number of steps e.g. for $x = 8$, it is needed to take $n = x - 2 = 8 - 2 = 6$, the value for the previous $x = 7$ must be known, i.e. $a_n = 64$ from Table 3, and substituting to the equation $a_{n+1} = 6 * (3 * 6 + 7) - 64 = 86$ which is the number of CA steps needed to calculate the result of 8^2 (as corresponds to the value from Table 3 for this CA for $x = 8$ observed in our CA simulator).

The aforementioned example also shows that it is not possible in some cases to express the number of CA steps for arbitrary $n \geq 1$ which means that the development of some CA for low values of x exhibit some anomaly in comparison with the development for larger input values. The reason for this behavior still remains in general an open question but our observations indicate that this is probably the case of CA exhibiting a complex (and mostly visually very attractive) pattern generated by the development. For a low input value (e.g. $x < 5$ in case of the CA–40–14 from Table 3) the development does not need to involve all possible state transitions before reaching the result state, which would otherwise emerge for larger x. Therefore, the development for $x < 5$ is specific, leading to the numbers of steps that is not possible to express together with the numbers of steps for larger input values.

The overall comparison of some selected CA is shown in Figure 6. As evident from this figure (and from the equations in Table 3 and 4), the computational efficiency of the CA (i.e. the number of steps needed for given x) in all cases

[1] https://www.wolframalpha.com/

14

Table 3. The number of steps of resulting CA needed to calculate the square for $x = 2, ..., 16$ together with expressions allowing to calculate the number of steps for given $n = x - 1$. (Part 1 of the CA comparison.)

the input value of x														
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
the equation derived for the num. of steps of a given CA and the num. of CA steps needed to finish the calculation of x^2														
CA–50–12: $a_{n+1} = n(3n + 7) - a_n$ for $n \geq 2$														
5	11	18	30	46	64	86	110	138	168	202	238	278	320	366
CA–30–08: $a_n = 2n^2 + n + 1,\ n \geq 1$														
4	11	22	37	56	79	106	137	172	211	254	301	352	407	466
CA–50–27: $a_n = n(2n + 3),\ n \geq 1$														
5	14	27	44	65	90	119	152	189	230	275	324	377	434	495
CA–40–01: $a_n = 2n^2 + 3n + 3,\ n \geq 1$														
8	17	30	47	68	93	122	155	192	233	278	327	380	437	498
CA–40–14: $a_{n+1} = \frac{a_n(n-3)}{n-5} - \frac{2(11n+13)}{n-5},\ n \geq 5$														
8	16	29	46	68	94	124	158	196	238	284	334	388	446	508
CA–30–11: $a_{n+1} = -a_n + 4n^2 + 13n + 11$ for $n \geq 2$														
3	17	35	51	76	100	133	165	206	246	295	343	400	456	521
CA–30–00: $a_{n+1} = \frac{a_n(n-1)}{n-3} + \frac{-15n-19}{n-3},\ n \geq 3$														
4	17	32	51	74	101	132	167	206	249	296	347	402	461	524
CA–50–22: $a_{n+1} = 2(2n^2 + 7n + 7) - a_n$ for $n \geq 1$														
14	24	34	58	76	108	134	174	208	256	298	354	404	468	526
CA–20–07: $a_n = \frac{1}{4}(8n^2 + 22n - 5(-1)^n - 3),\ n \geq 1$														
8	17	35	52	78	103	137	170	212	253	303	352	410	467	533

Table 4. The number of steps of resulting CA needed to calculate the square for $x = 2, ..., 16$ together with expressions allowing to calculate the number of steps for given $n = x - 1$. (Part 2 of the CA comparison.)

the input value of x														
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
the equation derived for the num. of steps of a given CA and the num. of CA steps needed to finish the calculation of x^2														
CA–40–34: $a_n = \frac{1}{4}(8n^2 + 22n - 3(-1)^n + 3),\ n \geq 1$														
9	19	36	54	79	105	138	172	213	255	304	354	411	469	534
CA–50–07: $a_n = \frac{1}{2}(5n^2 + 3n - 2),\ n \geq 1$														
3	12	26	45	69	98	132	171	215	264	318	377	441	510	584
CA–20–00: $a_{n+1} = -a_n + 5n^2 + 9n + 2$ for $n \geq 1$														
5	13	27	47	71	101	135	175	219	269	323	383	447	517	591
CA–30–31: $a_{n+1} = \frac{a_n(n^2-8n-2)}{n^2-10n+7} + \frac{-27n^2+13n+9}{n^2-10n+7},\ n \geq 9$														
2	16	33	54	79	108	141	178	219	271	331	397	469	547	631
CA–50–17: $a_n = 3n^2 + 1,\ n \geq 1$														
4	13	28	49	76	109	148	193	244	301	364	433	508	589	676
CA–20–27: $a_{n+1} = \frac{a_n(n-1)}{n-3} - \frac{15(n+1)}{n-3},\ n \geq 3$														
7	15	30	51	78	111	150	195	246	303	366	435	510	591	678
CA–50–15: $a_n = 3n^2 + 2n + 2,\ n \geq 1$														
7	18	35	58	87	122	163	210	263	322	387	458	535	618	707
Wolfram's CA: $a_n = 3n^2 + 7n + 2,\ n \geq 1$														
12	28	50	78	112	152	198	250	308	372	442	518	600	688	782

exhibit a quadratic form. However, the CA efficiency differ substantially between various solutions. Whilst Wolfram's CA exhibits the highest numbers of steps out of all CA that were available for the square calculations herein, our experiments showed that (1) this approach can be improved substantially and (2) various other solutions with a moderate efficiency exist for this task. Some of them are presented as visualisation of the appropriate CA development in Figure 7 and 8. Specifically, the CA–30–08 from Figure 7a represents an example of the second best result discovered from our evolutionary experiments. Its simple pattern exhibit a high degree of regularity which is probably the cause of very simple equation expressing the number of steps for given x (Table 3). More complex, less computationally efficient and visually interesting patterns are generated by CA–50–22 (Figure 7b, Table 3) and CA–40–34 (Figure 7c, Table 4). On the other hand, the CA–30–31 from Figure 8a produces results of calculating x^2 that is composed of various (stable) state values and this CA also exhibit the most complex equation needed to express its number of steps for given x (see Table 4). Together with CA from Figure 8b,c it belongs to the least computationally efficient (i.e. requires many steps to produce the result – see the comparison in Figure 6) but also can be viewed as solutions that demonstrate the variety of different styles of how the result of x^2 in CA can be achieved.

4.4 Discussion

In most cases of the experimental settings the EA was able to produce at least one generic solution for the CA-based square calculation. Despite the 2 million generation limit, the results from Table 1 and 2 show that the average number of generations is mostly below 1 million, which indicates a potential of the EA to efficiently explore the search space. In comparison with the initial study of this problem proposed in [5], where 200,000 generations were performed, the significant increase of this parameter herein is important with respect to achieving a reasonable success rate and producing generic solutions (note that an initial comparison of various ranges for x evaluated in the fitness was proposed in [5], the result of which was considered in this work).

As regards the RA-fitness, which can be considered as the main technique proposed herein for the evolution of cellular automata, a more detailed analysis was performed with various multi-state CA. As the results in Table 1 show that the number of generic solutions increases for the number of states from 4 to 8, then for 10-state CAs a significant reduction can be observed. This is probably caused by the exponential increase of the search space depending on the number of states. The results indicate that the 8-state setup represents a very feasible value that may be considered as sufficient for this kind of problem (note that 6 generic solutions were obtained for this setup).

In both sets of experiments with the RA-fitness and SR-fitness, a phenomenon of a reduction of the number of states was observed. This is possible due to the identification of just the rules that are needed for the CA development to calculate the square out of all the rules generated from the evolved CMR-based

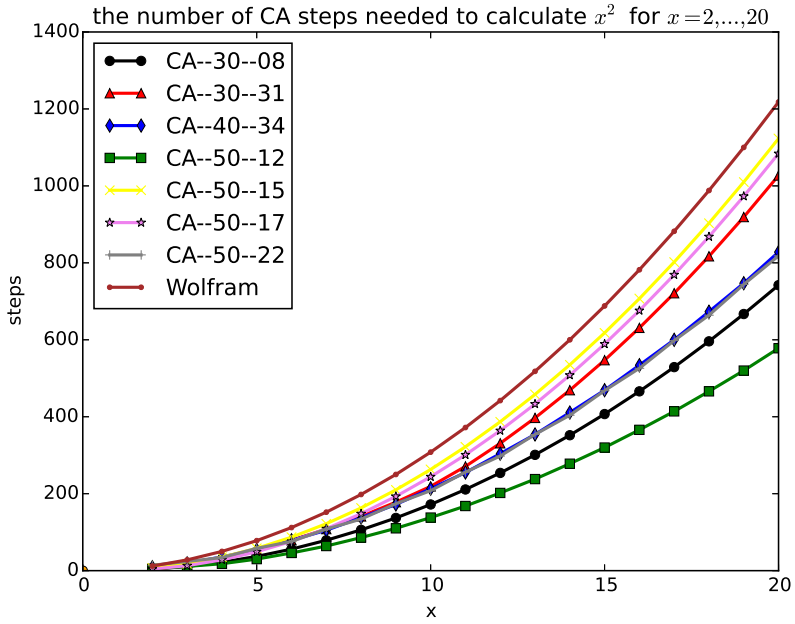
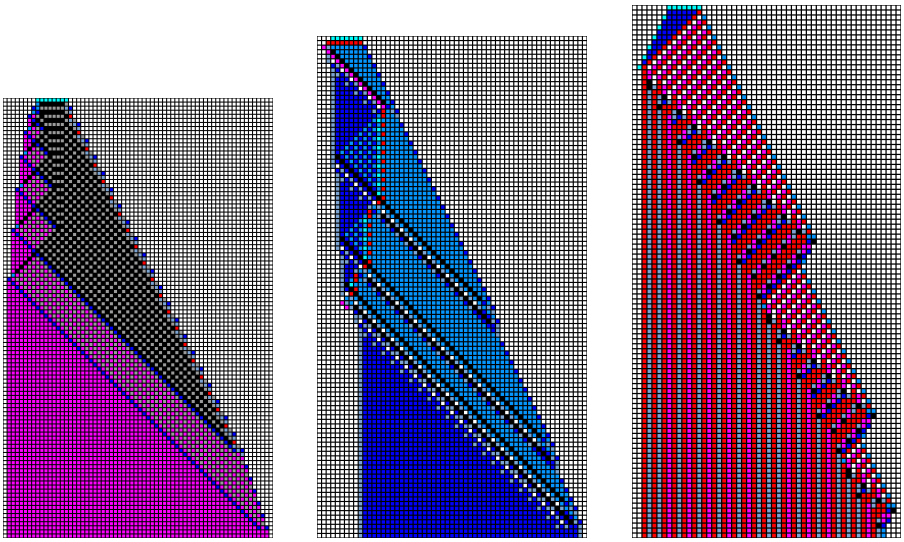


Fig. 6. Evaluation of the computational efficiency (i.e. the number of steps needed to achieve the result of x^2) of some selected CA whose development is also presented visually in various figures in this paper. A comparison with the existing Wolfram's CA [26] (the top-most curve) is included.



(a) (b) (c)

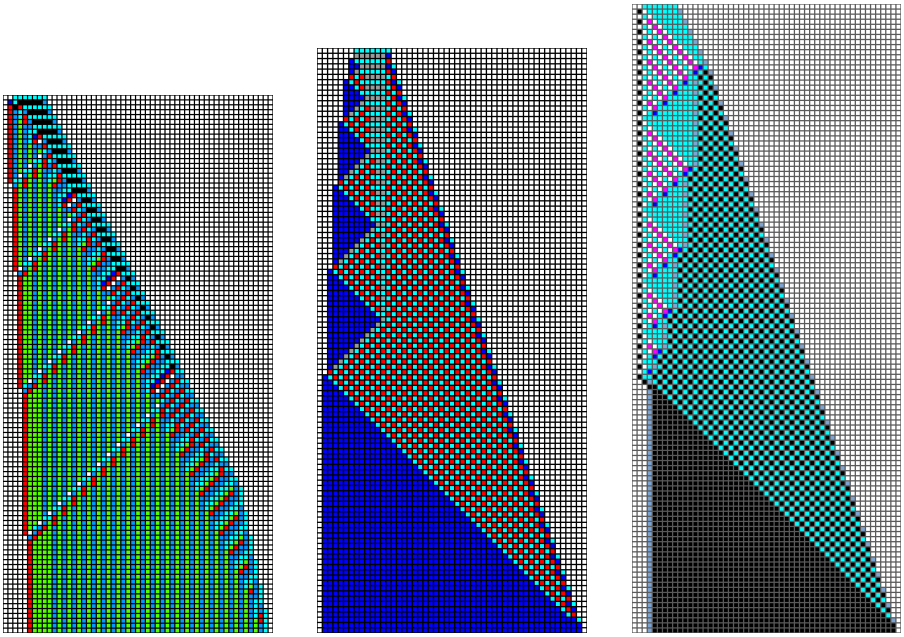
Fig. 7. Visualisation of the development of some selected squaring cellular automata obtained from our experiment: (a) CA–30–08, $x = 8$, (b) CA–50–22, $x = 7$, (c) CA–40–34, $x = 7$.

transition function for every valid combination of states in the cellular neighbourhood. It was determined that the CAs in some cases do not need all the available cell states to perform the given operation.

5 Evolution of Complex 2D Cellular Automata

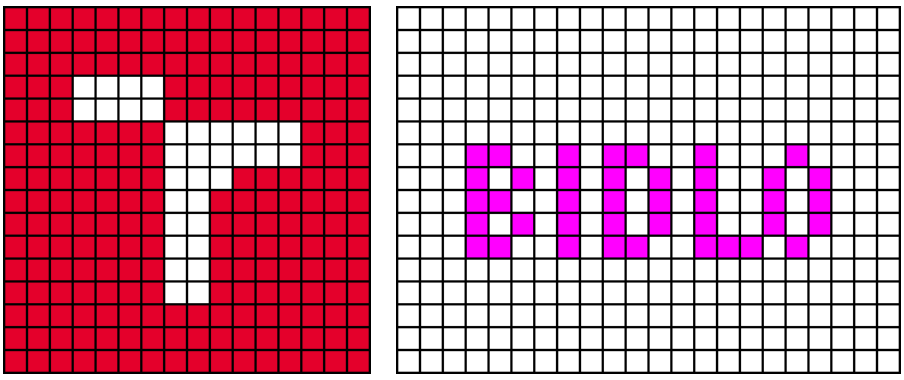
In order to provide a wider overview of what CA the proposed method can handle, some experiments dealing with the evolution of uniform multi-state 2D automata were conducted. The pattern development problem was chosen as a case study. In particular, some non-trivial and asymmetric patterns were chosen as shown in Figure 9.

In order to evaluate a candidate CA, up to 40 development steps were performed and the steps 16-40 was assigned a partial fitness calculated as the number of cells in correct states with respect to the given pattern. The final fitness of the candidate CA was the maximum of the partial fitness values. The reason for this setup is to reduce the time needed for the evaluation because the patterns probably cannot be finished in less than 16 steps (this values was determined empirically). Therefore, no inspection of cell states is performed in steps 1-15. Moreover, no exact number of steps is known in which a pattern can be finished so that the aforementioned range of steps provides the evolution with a wider



(a) (b) (c)

Fig. 8. Visualisation of the development of some selected squaring cellular automata obtained from our experiment: (a) CA–30–31, $x = 7$, (b) CA–50–17, $x = 7$, (c) CA–50–15, $x = 7$.



(a) (b)

Fig. 9. Samples of selected patterns treated in the experiments for the pattern development problem in 2D CA: (a) the BUT logo (note that the T-like structure, composed of cells in state 1, is the subject of the CA development on the red background – cells in state 0), (b) a label containing the author's surname (composed of cells in state 1, the other cells are in state 0).

space to discover a solution. This setup does not eliminate a possibility of destroying the pattern that emerged at a certain step during the subsequent CA development. However, the development of a stable pattern was not a primary goal of these experiments.

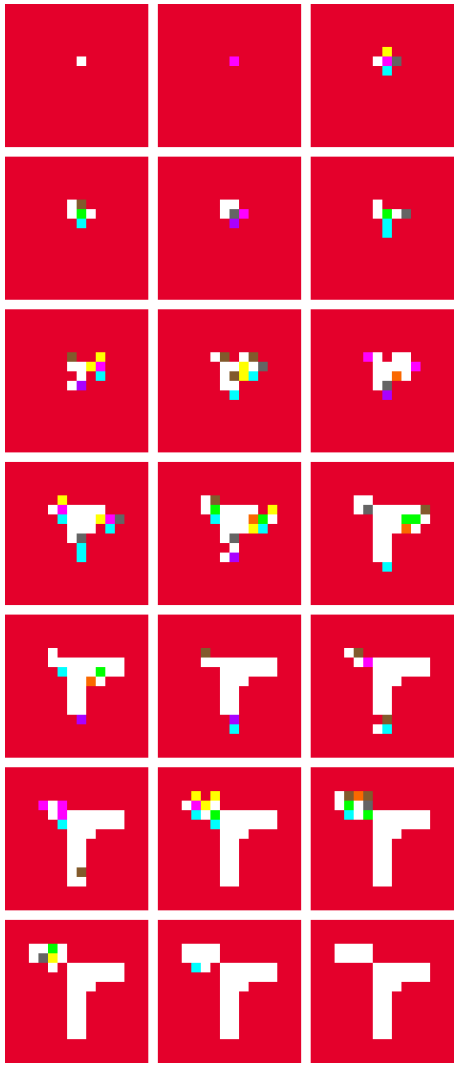
It is difficult to provide statistical data from these experiments since only very few successful results have been obtained so far and there has been a research in progress regarding the optimization of evolutionary techniques used for the design of complex cellular automata. Therefore, only some selected results are presented in this section.

The first experiment — the development of the BUT logo — dealt with the CA working with 10 cell states. There are 10^5 various combinations in the 5-cell neighborhood which implies 10^{100000} possible transition functions. A single state-1 cell (a seed) was used as an initial state of the CA. The evolution managed to find a transition function (in the form of a sequence of CMRs that was converted to the table-based representation for the presentation purposes) that successfully develops the initial seed into the given pattern as shown in Figure 9a. Although the pattern stability was not required, the pattern no longer changes by further CA development. A sample of the complete sequence of states leading to the emergence of this pattern (the BUT logo) is shown in Figure 10 together with the appropriate transition function. The CA uses 90 transition rules and needs 20 steps to finish the pattern.

The second experiment — the development of the author's surname — delat with the CA working with 12 cell states. There are 12^5 various combinations in the 5-cell neighborhood which implies 10^{248832} possible transition functions. In fact, this pattern requires to develop several (separate) structures (letters) of non-zero cells which form the complete label. A sample of the CA development together with the transition function is shown in Figure 11. This is the only result obtained so far which is able to fully develop this pattern (from a triples of separate cells in states 1, 2 and 3 as an initial CA state – see the top-left corner of Fig. 11a). Similarly to the previous experiment, this pattern is also stable during further CA development which is especially interesting due to its increased complexity. The CA needs 32 steps to develop the target label from the initial state and the transition function consists of 161 rules (see Figure 11b).

5.1 Discussion

Although the evolution of multi-state 2D CA is much more difficult than the 1D CA, the experiments provided some successful results with various target patterns. The results presented in this section probably represent the first case when complex 2D CA with at least 10 cell states were automatically designed using an evolutionary algorithm in a task of the non-trivial exact pattern development. In addition to the CA shown in Fig. 10 and 11, the evolution succeeded in searching other patterns as well (e.g. French flag, Czech flag, moving labels, replicating objects or multi-state gliders). This indicates that the proposed design method may be applicable in a wider area of cellular automata. A limitation for a higher success rate of the evolutionary experiments probably lies in the



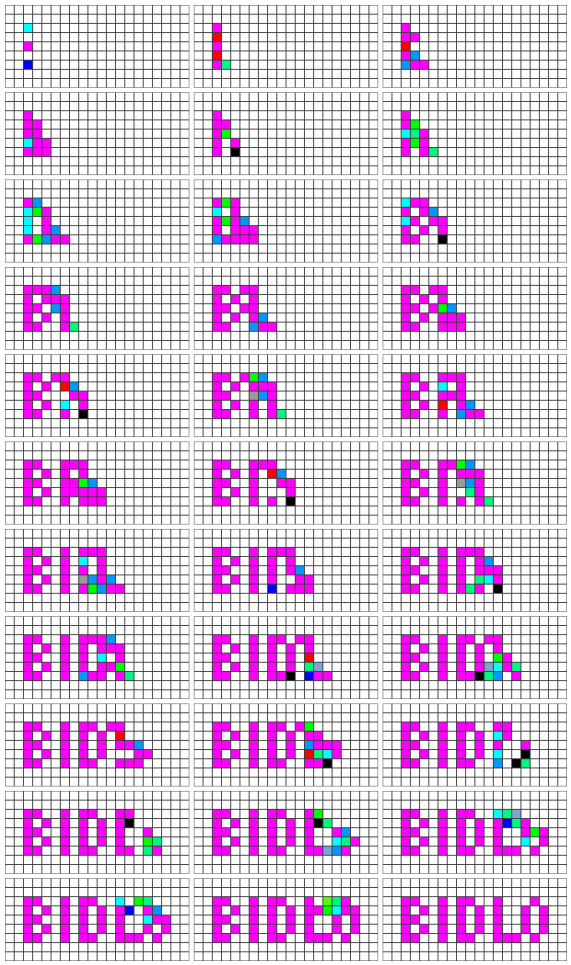
(a)

```
0 0 0 0 8 --> 3
0 0 0 0 9 --> 3
0 0 0 3 1 --> 1
0 0 0 3 3 --> 1
0 0 0 4 0 --> 1
0 0 0 6 0 --> 1
0 0 0 9 0 --> 1
0 0 1 0 0 --> 9
0 0 1 0 1 --> 4
0 0 1 4 0 --> 9
0 0 3 0 1 --> 4
0 0 3 0 9 --> 4
0 1 0 1 1 --> 1
0 1 0 3 2 --> 1
0 1 1 0 8 --> 0
0 1 1 1 0 --> 9
0 1 2 1 3 --> 1
0 1 2 1 6 --> 1
0 1 4 0 1 --> 1
0 1 4 0 2 --> 1
0 1 4 5 2 --> 1
0 1 8 0 0 --> 9
0 1 9 0 9 --> 1
0 2 1 0 0 --> 9
0 2 1 8 0 --> 3
0 3 0 3 3 --> 5
0 3 9 8 6 --> 2
0 4 0 0 9 --> 9
0 4 5 4 1 --> 2
0 5 4 0 8 --> 1
0 8 9 0 0 --> 1
0 9 0 0 0 --> 8
0 9 1 9 1 --> 3
0 9 3 1 1 --> 1
0 9 8 0 0 --> 1
1 0 0 1 1 --> 1
1 0 0 4 1 --> 1
1 0 1 0 7 --> 4
1 0 6 0 0 --> 7
1 0 6 1 0 --> 0
1 0 7 0 6 --> 4
1 1 1 5 0 --> 5
1 1 1 9 1 --> 3
1 1 2 1 1 --> 1
1 1 3 1 3 --> 1
```

```
1 1 3 9 0 --> 5
1 1 4 0 1 --> 1
1 1 4 3 1 --> 1
1 1 5 1 0 --> 1
1 1 5 2 3 --> 2
1 1 7 0 0 --> 1
1 1 8 0 1 --> 1
1 1 8 1 0 --> 1
1 1 8 3 0 --> 1
1 1 8 9 7 --> 2
1 2 2 1 1 --> 1
1 3 6 0 0 --> 1
1 5 1 0 0 --> 0
1 6 1 0 0 --> 0
2 0 6 0 0 --> 7
2 0 6 0 6 --> 1
2 0 6 1 0 --> 0
2 1 5 1 0 --> 1
2 3 6 0 0 --> 1
2 5 1 0 0 --> 0
2 8 3 1 1 --> 1
3 1 0 6 0 --> 3
3 1 9 1 6 --> 2
3 1 9 3 6 --> 2
3 1 9 8 6 --> 2
3 3 1 0 2 --> 8
3 3 9 0 6 --> 1
3 4 3 6 0 --> 5
4 1 1 0 6 --> 8
4 1 2 1 6 --> 8
4 1 6 0 0 --> 1
4 1 8 0 2 --> 1
5 1 3 6 0 --> 5
5 2 1 8 1 --> 3
6 0 6 0 0 --> 7
7 0 0 0 0 --> 6
8 0 0 0 0 --> 6
8 0 0 1 0 --> 6
8 0 6 0 6 --> 1
8 0 7 0 0 --> 6
8 1 2 1 0 --> 0
9 0 0 0 0 --> 6
9 0 0 1 0 --> 6
9 1 0 0 0 --> 6
9 1 9 1 6 --> 2
```

(b)

Fig. 10. Example of an evolved CA for the development of the BUT logo. The initial state consists of a single cell state 1, the other cells possess state 0. (a) The sequence of CA states producing the final pattern (ordered from left to right and top to bottom). (b) The appropriate transition rules for this CA.



(a)

```
0  0  1  0  0 --> 2
0  0  1  0  2 --> 2
0  0  1  2  2 --> 2
0  0 10 11  0 --> 8
0  0  2 10  0 --> 0
0  0  2 10  1 --> 1
0  0  2 10  2 --> 1
0  0  3  0  0 --> 2
0  0  5 11  0 --> 0
0  0  8 11 10 --> 0
0  1 11  4  3 --> 0
0 10  0  0 11 --> 4
0 10  0  0  2 --> 2
0 10  6  0  2 --> 2
0 11  0  0  0 --> 2
0 11  0  0  2 --> 2
0 11  0  0  6 --> 2
0 11  4  0 11 --> 10
0 11  4  0  2 --> 2
0  2  0  0 10 --> 6
0  2  0  0 11 --> 6
0  2  0  0  6 --> 6
0  2  0 11  0 --> 4
0  2 10  0  2 --> 2
0  2 10  0  5 --> 11
0  2 10  2  0 --> 2
0  2 10  6  0 --> 2
0  2 10  6  2 --> 2
0  2 11  0  0 --> 2
0  2 11  0  2 --> 2
0  2 11  2  0 --> 2
0  2  2  0 10 --> 10
0  2  2  0  5 --> 10
0  2  2  0  7 --> 10
0  2  2 10  0 --> 0
0  2  2 10  2 --> 0
0  2  2  2  1 --> 0
0  2  2  6  2 --> 0
0  2  3  0  0 --> 2
0  2  4  1 11 --> 0
0  2  4  6  0 --> 2
0  2  6  0 10 --> 10
0  2  6  0 11 --> 10
0  2  6  0  2 --> 2
0  3  0  0  0 --> 11
0  3  0  2  0 --> 11
0  4  0  0  2 --> 11
0  5  0  0  0 --> 11
0  5  0  3  0 --> 11
0  6  0  0  2 --> 2
0  6  0  5  0 --> 2
0  7  0  0  0 --> 2
0  7  0  0 11 --> 6
0  7  0  0  2 --> 2
0  7  0  0  4 --> 2
0  7  6  0  2 --> 2
0  8 11  2  1 --> 0
0  9  6  2 10 --> 0
1  0  0  0  2 --> 7
1  0  1  0  1 --> 2
1  0  1  0  2 --> 2
1  0  2 10  0 --> 6
1  0  2 10  2 --> 1
1  0  6  0  0 --> 2
1 11  6  0  0 --> 2
1  4  6  2  0 --> 0
10  0 11  2  0 --> 6
10  0  2 11  0 --> 3
10  0  2  2  2 --> 1
10  0  2  2  6 --> 1
10  0  2  2  9 --> 1
10  1  0  2  0 --> 10
10  1 11  2 10 --> 0
10  2  0  2  0 --> 10
10  2  2  2  2 --> 0
10  2  5 11  0 --> 3
10  3  0  2  0 --> 10
10  4  1  2  6 --> 0
11  0  2  6  1 --> 1
11  0  3  2  0 --> 6
11 10  1  2  2 --> 2
```

```
11  2  0  2  0 --> 10
11  2 10  2  0 --> 0
11  2  2  5  0 --> 0
 2  0  0  0  2 --> 7
 2  0  0  0  3 --> 7
 2  0  0  2  0 --> 11
 2  0  0  2  2 --> 11
 2  0  1  0  0 --> 2
 2  0  1  0  2 --> 2
 2  0  1  0  6 --> 2
 2  0  1  2  2 --> 2
 2  0 10 11 11 --> 1
 2  0 10  2  1 --> 2
 2  0 11  1  2 --> 2
 2  0  2  0 10 --> 7
 2  0  2 10  1 --> 1
 2  0  2 10  2 --> 1
 2  0  2 11  0 --> 6
 2  0  3  0  0 --> 2
 2  0  5  0  0 --> 2
 2  0  5  0 11 --> 10
 2  0  6  2  0 --> 2
 2  0  6  2  7 --> 2
 2  0  7  0 11 --> 10
 2  0  7  0  2 --> 2
 2  0  7  6  0 --> 2
 2  0  9  6  0 --> 2
 2  0  9  6  2 --> 2
 2  1  2  0  0 --> 5
 2  1  2  0  5 --> 10
 2  1  2 11  0 --> 0
 2  1  2  2  0 --> 0
 2  1  2  2  2 --> 0
 2 10  2  6  2 --> 0
 2 10  6  2  0 --> 2
 2 11  1  2  0 --> 2
 2 11  1  2  5 --> 0
 2  2 10  0  0 --> 11
 2  2 10  0  2 --> 11
 2  2 10  6  2 --> 0
 2  2  2  0  0 --> 5
 2  2  2  2  0 --> 10
 2  2  2  2  2 --> 0
 2  2  5 11  0 --> 0
 2  5  0  0  0 --> 11
 2  6  0  2  0 --> 2
 2  6  2  2 11 --> 0
 2  6  2  6  6 --> 0
 2  7  0  0  6 --> 2
 2  7  0  2  0 --> 2
 2  7 11  1  2 --> 0
 2  9  6  2  0 --> 2
 2  9  6  2 11 --> 0
 4  3 11  2  0 --> 0
 4  3  2  2  0 --> 0
 4  5 11  6  0 --> 0
 6  0 11  2  0 --> 6
 6  0  2  2  0 --> 6
 6  0  2  2  2 --> 1
 6  0  7 11  2 --> 1
 6  1 10  2  0 --> 0
 6  1 11  2  2 --> 0
 6  1  2  2  0 --> 0
 6  2 10  6  0 --> 0
 6  2  2  2  0 --> 0
 6  2  2  2  1 --> 0
 6  2  2  2  2 --> 0
 6  2  2  2  6 --> 0
 7  0  0  2  0 --> 9
 7  0  0  2  1 --> 9
 7  0 11  4  3 --> 1
 7  0  2  0  7 --> 7
 7  0  2 11  0 --> 6
 7  0  2  2  0 --> 6
 7  0  2  6  6 --> 1
 8  2 10  1  0 --> 0
 9  0  0 11  2 --> 9
 9  0  2  0  2 --> 7
 9  0  2 10  0 --> 3
```

(b)

Fig. 11. Example of an evolved CA for the development of the author's surname. The initial state consists of three vertically alligned cells in states 1, 2 and 3 respectively with a single state-0 cells between them. (a) The sequence of CA states producing the final "BIDLO" pattern (ordered from left to right and top to bottom). (b) The appropriate transition rules for this CA.

requirement of an exact pattern development. Our initial experiments suggest that promising areas for the CA applications may be those where approximate results are acceptable or the CA states allow us to tolerate some variations during the development. For example, the image processing, traffic prediction or design of approximative algorithms represent possible topics. Therefore, the future research will include modeling, simulation and optimization of such kinds of systems in cellular automata.

Even though the representation of the transition functions by means of conditional rules has proven a good applicability on various tasks, the optimization of the evolutionary algorithm used to search for the rules is probably still possible. This could not only improve the success rate of the evolutionary experiments but also reduce the computational effort and allow applying the concept of uniform computing platforms in real-world applications (e.g. with acceleration of the computations using modern reconfigurable technology).

6 Conclusions

In this study we have presented some advanced topics related to the evolution of complex multi-state cellular automata. In particular, an analysis of CA for the generic square calculations has been proposed in the first case study. The results showed that some various algorithms to perform this task exist in CA, which differ both in the complexity of resulting transition functions and the efficiency of the computation (i.e. the number of steps of the CA needed to produce the result). Moreover, our best results presented herein have overcome the known solution (Wolfram's squaring CA), providing a reduction of the number of steps by approximately 50%.

The second case study has dealt with the non-trivial pattern development problem in two-dimensional CA. Several results have been presented that provide an exact and stable pattern developed from a simple initial CA state. Cellular automata working with 10 and 12 cell states have been treated, which induce search spaces of enormous sizes. Despite low success rates of the evolution, the results obtained have shown that the automatic design of such CA is possible even though our ongoing experiments indicate that the evolutionary algorithm still provides a space for further optimization.

In general, the proposed results probably represent the first case of the automatic design of exact behaviour in CA with more than 10 cell states. We believe that these pieces of knowledge will allow us to further improve our design method and to apply cellular automata for modeling, simulation and optimization of real-world problems.

24

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II), project IT4Innovations excellence in science – LQ1602, and from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070".

References

1. Basanta, D., Bentley, P., Miodownik, M., Holm, E.: Evolving cellular automata to grow microstructures. In: Genetic Programming, Lecture Notes in Computer Science, vol. 2610, pp. 1–10. Springer Berlin Heidelberg (2003)
2. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for Your Mathematical Plays, 2nd Ed., Volume 4. A K Peters/CRC Press (2004)
3. Bidlo, M.: Investigation of replicating tiles in cellular automata designed by evolution using conditionally matching rules. In: 2015 IEEE International Conference on Evolvable Systems (ICES). pp. 1506–1513. Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE Computational Intelligence Society (2015)
4. Bidlo, M.: Evolution of generic square calculations in cellular automata. In: Proceedings of the 8th International Joint Conference on Computational Intelligence - Volume 3: ECTA. pp. 94–102. SciTePress - Science and Technology Publications (2016)
5. Bidlo, M.: On routine evolution of complex cellular automata. IEEE Transactions on Evolutionary Computation 20(5), 742–754 (2016)
6. Bidlo, M., Vasicek, Z.: Evolution of cellular automata with conditionally matching rules. In: 2013 IEEE Congress on Evolutionary Computation (CEC 2013). pp. 1178–1185. IEEE Computer Society (2013)
7. Codd, E.F.: Cellular Automata. Academic Press, New York (1968)
8. Durand, B., Rka, Z.: The game of life: Universality revisited. In: Mathematics and Its Applications, Volume 460 Cellular Automata. pp. 51–74. Springer Netherlands (1999)
9. Elmenreich, W., Fehérvári, I.: Evolving self-organizing cellular automata based on neural network genotypes. In: Proceedings of the 5th International Conference on Self-organizing Systems. pp. 16–25. Springer (2011)
10. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
11. Ilachinski, A.: Cellular Automata: A Discrete Universe. World Scientific (2001)
12. Jean-Yves Perrier, Moshe Sipper, J.Z.: Toward a viable, self-reproducing universal computer. Physica D 97(4), 335–352 (1996)
13. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Systems 4(3), 299–318 (1990)
14. Mardiris, V., Sirakoulis, G., Karafyllidis, I.: Automated design architecture for 1-d cellular automata using quantum cellular automata. Computers, IEEE Transactions on 64(9), 2476–2489 (2015)
15. von Neumann, J.: The Theory of Self-Reproducing Automata. A. W. Burks (ed.), University of Illinois Press (1966)

16. Ninagawa, S.: Solving the parity problem with rule 60 in array size of the power of two. Journal of Cellular Automata 8(3–4), 189–203 (2013)
17. Rendell, P.: A universal turing machine in conway's game of life. In: 2011 International Conference on High Performance Computing and Simulation (HPCS). pp. 764–772 (2011)
18. Rendell, P.: A fully universal turing machine in Conway's game of life. Journal of Cellular Automata 9(1–2), 19–358 (2013)
19. Sahoo, S., Choudhury, P.P., Pal, A., Nayak, B.K.: Solutions on 1-d and 2-d density classification problem using programmable cellular automata. Journal of Cellular Automata 9(1), 59–88 (2014)
20. Sahu, S., Oono, H., Ghosh, S., Bandyopadhyay, A., Fujita, D., Peper, F., Isokawa, T., Pati, R.: Molecular implementations of cellular automata. In: Cellular Automata for Research and Industry. pp. 650–659. Lecture Notes in Computer Science, Vol. 6350, Springer (2010)
21. Sipper, M.: Quasi-uniform computation-universal cellular cutomata. In: Advances in Artificial Life, ECAL 1995, Lecture Notes in Computer Science, Vol. 929. pp. 544–554. Springer Berlin Heidelberg (1995)
22. Sridharan, K., Pudi, V.: Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology. Springer International Publishing Switzerland (2015)
23. Stefano, G.D., Navarra, A.: Scintillae: How to approach computing systems by means of cellular automata. In: Cellular Automata for Research and Industry. pp. 534–543. Lecture Notes in Computer Science, Vol. 7495, Springer (2012)
24. Suzudo, T.: Searching for pattern-forming asynchronous cellular automata – an evolutionary approach. In: Cellular Automata, Lecture Notes in Computer Science, vol. 3305, pp. 151–160. Springer Berlin Heidelberg (2004)
25. Tempesti, G.: A new self-reproducing cellular automaton capable of construction and computation. In: Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life. pp. 555–563. Lecture Notes in Artificial Intelligence, Vol. 929, Springer (1995)
26. Wolfram, S.: A New Kind of Science. Wolfram Media, Champaign IL (2002)
27. Yuns, J.B.: Achieving universal computations on one-dimensional cellular automata. In: Cellular Automata for Research and Industry. pp. 660–669. Lecture Notes in Computer Science Volume 6350, Springer (2010)

Appendix VII

Evolution of Cellular Automata Development Using Various Representations

BIDLO Michal

In: *GECCO'19 Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Praha: Association for Computing Machinery, 2019, pp. 107-108. ISBN 978-1-4503-6748-6. Accessible online: https://dl.acm.org/citation.cfm?id=3321881

Evolution of Cellular Automata Development Using Various Representations

Michal Bidlo
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
bidlom@fit.vutbr.cz

ABSTRACT

This paper introduces a comparative summary regarding an evolution of multi-state cellular automata by means of various representations of their transition functions. In particular, a conventional table-based representation and an advanced approach using so-called Conditionally Matching Rules is applied. The French flag development from a seed is considered as a case study. The results show some remarkable differences in the cellular automata behaviour that are evidently caused by the representation used. They include the issue of emergence of the pattern from a chaotic state or rather a more systematic construction, a stability of the pattern developed and a limitation of its successful construction to fixed-size automata only. The comparison of the results is enabled by using a custom variant of genetic algorithm that provided working solutions for both representations of the transition function.

CCS CONCEPTS

• **Mathematics of computing** → **Evolutionary algorithms**; **Developmental representations**;

KEYWORDS

cellular automaton; transition function; development; evolutionary algorithm

ACM Reference Format:
Michal Bidlo. 2019. Evolution of Cellular Automata Development Using Various Representations. In *Proceedings of (GECCO '19 Companion).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321881

1 INTRODUCTION

Cellular automata (CAs) have been introduced in [5] as a mathematical model of complex systems in which the space, time and states are discrete. A cell represents a basic CA element, typically arranged in a regular lattice, whose state develops in steps according to a local transition function. Although CAs have extensively been studied both theoretically (e.g. [6], [1]) and practically (e.g. designing nano-scale arithmetic circuits [4] or performing computations [3]), the design of the transition function in order to achieve a given (global) CA behaviour (i.e. a solution to a given task) represents a difficult problem because this behaviour *emerges* from local interactions between many cells and is hard to predict it from the transition function itself. Therefore, the aim is to automate this process, e.g. using evolutionary algorithms (EAs) to design CAs.

In this paper we mention some approaches that may be utilised for the evolutionary design of CA and compare them with respect to the properties of the solutions obtained. Specifically, two representation techniques of the CA transition functions will be considered: (a) a conventional table-based representation and (b)a method called Conditionally Matching Rules (CMRs). Whilst the conventional approach utilises a set of rules, each of which determines a new state of a cell for a given valid combination of states in its neighbourhood (typically in the form $N\ W\ C\ E\ S\ \rightarrow\ C_{new}$, where C represents the state of the (*C*entral) cell to be updated and N, W, E and S is the state of its *N*orth, *W*est, *E*ast and *S*outh neighbour, respectively, and C_{new} is the new cell state), the CMRs use more general relations between the cell states and values specified in the CMRs as described in Section 2.

The goal of this paper is to show some initial results that indicate that both aforementioned representation techniques can be applied for the evolution of complex 2D CA but the latter may provide remarkably more robust and efficient solutions.

2 CONDITIONALLY MATCHING RULES

The concept of Conditionally Matching Rules and their evolutionary design is described in detail in [2]. For the purposes of this paper, let us consider a 2D CA working with 5-cell neighbourhood (including the *N*orth, *W*est, *C*entral, *E*ast and *S*outh cell). A conditional rule for such CA is defined as $(cnd_N\ s_N)\ (cnd_W\ s_W)\ (cnd_C\ s_C)\ (cnd_E\ s_E)\ (cnd_S\ s_S) \rightarrow s_{new}$, where cnd_x denote a condition operator ($=$, $\neq$, $\leq$ or $\geq$) and s_y represent a state value. Each CMR contains a pair *(a condition and a state value)* that corresponds to (is evaluated with respect to) a specific cell in the neighbourhood. A finite sequence of CMRs represents a transition function that, for example, contains a rule $(\neq 1)(\neq 2)(\leq 1)(\geq 2)(= 3) \rightarrow 2$. Let c_N, c_W, c_C, c_E, c_S be cells in states $2, 3, 0, 3, 3$ respectively, and a new state of c_C needs to be determined. The CMRs are evaluated sequentially until a rule is found whose all conditions are true with respect to the cell states in the given neighbourhood. According to the aforementioned rule, $c_N \neq 1$ is true as $2 \neq 1$, similarly $c_W \neq 2$ is true $(3 \neq 2)$ and the remaining conditions are also true. Therefore, this CMR is said to match, i.e. $s_{new} = 2$ on its right side will be the new state of c_C. If no matching rule is found, then the cell keeps its current state. Note that the CMR-based transition functions can be transformed to the appropriate table rules which preserves the original concept of CAs [2].

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321881

Table 1: Successful runs (out of 100) of our custom EA generating 16 offspring from the best half of 8 individuals.

#states	6	8	10	12	14	16
Table repr.	0	1	7	4	1	0
40 CMRs	0	0	0	0	0	1
50 CMRs	1	0	2	5	3	2
60 CMRs	11	15	18	14	14	15
70 CMRs	19	21	34	39	43	36

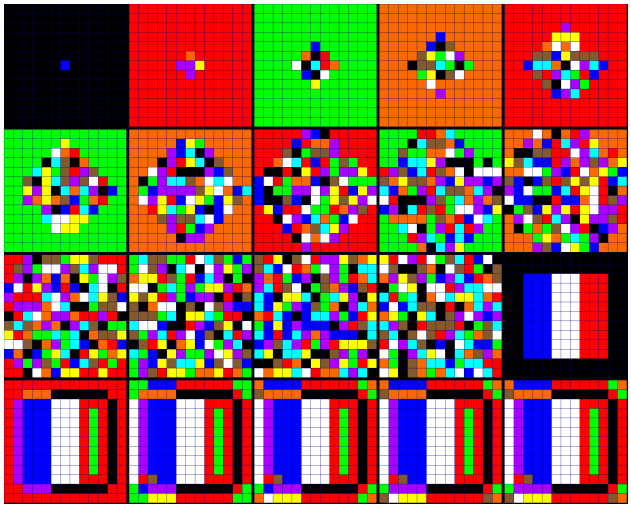


Figure 1: The French flag solution evolved using the conventional table-based approach in a CA working with 10 states[1].

3 EXPERIMENTAL RESULTS

We are tuning several evolutionary techniques which has shown necessary for obtaining working results. For example, the standard genetic algorithm failed in all cases of experiments we conducted so far. Therefore, a custom EA was applied whose preliminary results (the success rates of the CA evolution using both the table-based representation and the CMR method) are shown in Table 1.

Two samples of resulting CA development are shown in Fig. 1 (for the table representation) and Fig. 2 (for the CMR representation). As evident, the development of both CAs is different despite the fact that the CA concept is the same (the evolved CMRs were converted to the equivalent table rules). It was determined that the table-based solution does not work in CA of other sizes, the emergence of the pattern is rather chaotic and at a single step only, then it is destroyed and never restored again.. The CMR solution can develop a stable pattern that is independent on the CA size, the construction of which is progressive (and more systematic) from the initial seed. The evolved solution from Fig. 1 utilises 1274 active rules (i.e. those changing the cell state) whilst the converted CMR solution from Fig. 2 works with 129 rules only. Note that we observed such remarkably distinguished features in all results obtained for various patterns in CAs working with various numbers of states using the two representations. The evolution of CMRs also exhibits significantly higher success rates than the table-based representation.

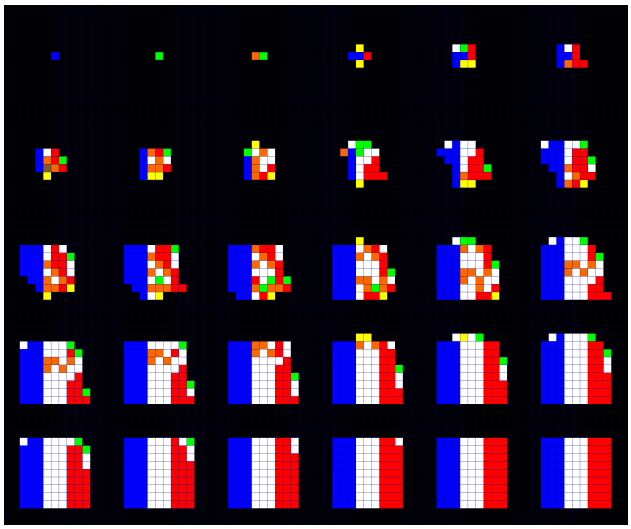


Figure 2: The French flag solution evolved using the CMR representation in a CA working with 8 states[1].

4 CONCLUSIONS

The results obtained may definitely be important for the future research of CAs since they clearly show how the representation influence not only the process of evolution but the also the form and features of the results. We also identified the importance of searching new variants of EAs which allowed us to design complex multi-state CAs that have not been obtained before. Therefore, the detailed statistical evaluation of the EA used, analysis of the results and solution of other problems in CAs represent our main work for the future research. We believe that a systematic study in this area may provide a better understanding of complex systems and help us optimise their automatic design using evolutionary techniques.

ACKNOWLEDGMENTS

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project LTC18053 and Large Infrastructures for Research, Experimental Development and Innovations project LM2015070 of the IT4Innovations National Supercomputing Center.

REFERENCES

[1] Andrew Adamatzky. 2010. *Game of Life Cellular Automata* (1st ed.). Springer Publishing Company, Incorporated.
[2] Michal Bidlo. 2016. On Routine Evolution of Complex Cellular Automata. *IEEE Transactions on Evolutionary Comp.* 20, 5 (2016), 742–754.
[3] Genaro J. Martínez, Andrew Adamatzky, and Harold V. McIntosh. 2017. *A Computation in a Cellular Automaton Collider Rule 110.* Springer International Publishing, Cham, 391–428.
[4] K. Sridharan and V. Pudi. 2015. *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology.* Springer International Publishing Switzerland.
[5] J. von Neumann. 1966. *The Theory of Self-Reproducing Automata.* A. W. Burks (ed.), University of Illinois Press.
[6] S. Wolfram. 2002. *A New Kind of Science.* Wolfram Media, Champaign IL.

[1] See an interactive simulator with this CA: https://github.com/bidlom/GECCO2019

Appendix VIII

Evolution of Cellular Automata with Conditionally Matching Rules for Image Filtering

BIDLO Michal

In: *2020 IEEE Congress on Evolutionary Computation (CEC 2020)*. Los Alamitos: IEEE Computational Intelligence Society, 2020, ISBN 978-1-7281-6929-3. Accessible online: https://ieeexplore.ieee.org/document/9185767

Conference CORE rank in the year of publication: B

Evolution of Cellular Automata with Conditionally Matching Rules for Image Filtering

Michal Bidlo
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, Brno, Czech Republic
bidlom@fit.vutbr.cz

***Abstract*—We present an evolutionary method for the design of image filters using two-dimensional uniform cellular automata. Specifically, a technique called Conditionally Matching Rules is applied to represent transition functions for cellular automata working with 256 cell states. This approach allows reducing the length of chromosomes for the evolution substantially which was a need for such high number of states since the traditional table-based encoding would require enormous memory space. The problem of removing Salt-and-Pepper noise from 8-bit grayscale images is considered as a case study. A cellular automaton will be initialised by the values of pixels of a corrupted image and a variant of Evolution Strategy will be applied for the design of a suitable transition function that is able to eliminate the noise from the image during ordinary development of the cellular automaton. We show that using only 5-cell neighbourhood of the cellular automaton in combination with conditionally matching rules the resulting filters are able to provide a very good output quality and are comparable with several existing solutions that require more resources. Moreover, the proposed evolutionary method exhibits a high performance which allows us to design filters in very short time even on a common PC.**

***Index Terms*—Evolution strategy, cellular automaton, conditional rule, image filter, salt-and-pepper noise.**

I. Introduction

Noise elimination from digital images represents a typical low level image processing task [1]. It often represents a significant step in the image pre-processing before performing advanced algorithms such as image segmentation, recognition or classification. Usually the noise elimination has to be implemented by means of non-linear functions (referred to as non-linear image filters) because the noise is inherently non-linear. Therefore, it is not possible to apply mathematical theories known from linear filters, which leads to mostly experimental design of non-linear filters.

This work was supported by Czech Science Foundation project GA19-10137S, by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602" and by the IT4Innovations infrastructure which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project IT4Innovations National Supercomputing Center - LM2015070".

A. Conventional image filter design

In conventional approaches, the image filter typically consists of two parts: a noise detector and a filtering algorithm. In such case, the filtering algorithm is executed only if the noise detector evaluates a given part of image as a noise. Otherwise, the input pixel is sent to the filter output unmodified. This approach allows decomposing the filter design into simpler parts which are then solved separately [1]. On the other hand, some approaches consider the filter as a "black box" that takes image data as inputs, implements a suitable function for the noise elimination and produces a filtered pixel as its output. This approach showed to be very powerful for evolutionary image filter design which will be mentioned in Section I-B. Conventional non-linear image filters of this kind are mostly based on the calculation of median out of the pixel values belonging to a given *filtering window* (usually of size 3x3 or 5x5 pixels). These techniques may be used for comparison with advanced (experimentally designed) filters.

The basic *median filter* (MF) is a special case of order statistic filters [2]. They can be implemented effectively in hardware using the concept of sorting networks as described in [3]. The *weighted median filter* (WMF) represents an extension of this concept, which assigns weights to some values within the filtering window. The *center weighted median filter* (CWMF) [4] is a special case which weights only the central value of the window.

The *adaptive median filter* (AMF) can be considered as a multi-level order statistic filter [5], the goal of which is to detect and subsequently replace corrupted pixels only. Usually, two levels working with the 3×3 and 5×5 filtering window, respectively, are utilised. The filtering is performed in two-phases. In the first phase, a sorting network is used for calculating the minimum, maximum and median value for the pixels inside the filtering window. In the second phase, these values are used together with the value of the original central pixel to decide whether it is affected by noise.

Another MF modification includes the *directional weighted median filter* (DWMF) [6] which utilizes a noise detector

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

that is based on the differences between the central pixel and its neighbours aligned with four main directions (horizontal, vertical and two diagonal). The noise detection is based on calculating the weighted sum of absolute differences between the value of the central pixel and pixels within the given direction and a threshold is used to determine whether the pixel is corrupted by noise and needs to be replaced. This method is applied iteratively (in 5–10 steps) with a decreasing threshold in order to achieve a required output quality.

Finally, let us mention the *pixel-wise MAD* (PWMAD) [7] as an iterative non-linear image filter that addresses the problem of random valued shot noise removal. The noisy pixels are detected using a local variance estimator based on the iterative calculation of the median of the absolute deviations from the median.

There are also other filtering principles that do not strictly utilise the notion of locality (filtering windows), use more complex operations and are therefore suitable for software implementations only (e.g. [1], [8]–[11]).

B. Design of image filters by means of evolutionary algorithms

Evolutionary algorithms [12] have recently been applied in many application domains including image processing. The evolutionary design of image filtering circuits can be found in the area of evolvable hardware methods [13]. Earlier works have dealt with several basic approaches, including the optimization of filter coefficients [14], [15] and stack filter evolution [16]. Currently, the design of image filters at the circuit level is usually performed by means of *Evolution Strategy* using the *Cartesian Genetic Programming* (CGP) representation [17]. A survey of image filter design methods that utilize CGP can be found in [18].

In addition to relatively simple non-linear functions composed of operations such as minimum, maximum, or average [19]–[21], advanced concepts have been investigated, e.g. using *bank of filters* [22], switching concept [23], [24] or fault-tolerant image filter design [25]. Later, the image filter evolution was accelerated using Field Programmable Gate Arrays (FPGAs) [26], [27], graphic processing units (GPU) [28], computer clusters and advanced pre-compilation techniques [29]. An advanced extensive study of evolutionary design of switching filters by means of CGP and their comparison with conventional filters can be found in [30].

C. Image filtering using cellular automata

Cellular automata (CA), originally introduced by John von Neumann in [31], represent a massively parallel computational platform suitable for various applications. The basic structure of a cellular automaton assumes a regular structure of *cells*, each of which at a given moment occurs in a *state* from a finite set of states. The behaviour (or *development*) of a CA is considered as a *synchronous* update of all cells according to a *transition function* in discrete time steps. The transition function determines the next state of a cell depending on the combination of states in its neighbourhood. For the purposes of this paper, *uniform two-dimensional (2D) CA* will be considered in which the cells are arranged into a square lattice and there is a *single* transition function according to which all cells are to be updated. The cellular neighbourhood will be defined uniformly for each cell and will consist of a given cell and its immediate neighbours in the north, south, east and west direction (it is a case of 5-cell neighbourhood also called von Neumann neighbourhood). The task of designing a CA for a given task consists of finding a suitable transition function (possibly with appropriate initial states of cells) in order to achieve a given behaviour performed by the CA development.

Cellular automata have also been applied to image processing tasks (including image and video compression, image resizing, edge detections and others). A wider overview can be found in [32]. Rosin applied a sequential floating forward search method for feature selection and identification of good rule sets for a range of tasks, namely noise filtering (also applied to grayscale images using threshold decomposition), thinning, and convex hulls [33]. Selvapeter and Hordijk introduced several modifications to the standard CA concept and applied them to improve the filtering performance. For example, a random CA rule solves the noise propagation present in deterministic CA filters. A mirrored CA is used to solve the fixed boundary problem. The authors showed that these methods can outperform some standard filtering methods [34]. Diosan et al. investigated various methods suitable for image segmentation by means of CA [35]. Kundra et al. applied CA to detect edges in digital images. The approach is based on 2-fold symmetry that implies 51 patterns from which the best rule for image detection can be derived [36].

The utilisation of the pure CA concept for image processing represents a difficult problem. The existing works are mostly limited to simplified cases (e.g. binary images only) or use the CA as a mean to derive suitable steps in order to perform the image processing algorithm. In this paper we apply basic CA whose state will directly represent an image and use an evolutionary algorithm in order to design a transition function (the filtering algorithm) as described in Section I-D.

D. Goal of this paper

In this paper we use a method for efficient representation of transition functions of CA, called Conditionally Matching Rules (CMRs) [37], and apply a variant of Evolution Strategy (ES) in order to find various transition functions suitable for removing Salt-and-Pepper noise from 8-bit grayscale images. The CA states will directly be represented by 256 possible pixel values of the filtered image. The filtering algorithm will be represented by a single transition function of the CA designed by the ES. The main objective is to demonstrate that using the 5-cell neighbourhood (instead of 3×3 or 5×5 windows) and CMR-based transition functions with only basic relational operators (specifically $=, \neq, \leq, \geq$), which represents significantly reduced resources against many other methods, high-quality image filters can be obtained. Moreover, we show that this approach in combination with the evolution strategy exhibits a very good performance which allows designing filters in very short time even on a common PC.

II. Evolution of cellular automata using conditionally matching rules

Conventionally, the local transition function of a CA is represented by a *table* specifying *transition rules* for any possible combination of states in the cellular neighbourhood. For the concept of 2D CA, introduced in Section I-C, each row of the table contains a rule of the form $s_N\ s_W\ s_C\ s_E\ s_S \rightarrow s_C^{new}$, where s_x is the state of a cell at position x in cellular neighbourhood and s_C^{new} is a new state of the cell in the middle of the neighbourhood for the next time step. However, the size of the table grows exponentially depending on the number of cell states and the size of the cellular neighbourhood which makes the representation and design of complex CA very difficult. Although a subset of rules could be specified to represent a transition function, the problem is how to identify the rules suitable for a given task. In this paper we deal with 256 cell states for which there are in total 256^5 possible rules. Designing a complete transition function in the form of the table for such CA is practically impossible because because its storage would require 1 TB of memory (considering a single rule to be encoded as the next state value represented as 1 byte).

A. Conditionally Matching rules (CMRs)

Conditionally matching rules can be considered as an advanced representation method for transition functions of complex CA. The first extensive study of this concept and its suitability for the evolutionary design of complex CA was proposed in [37] and recently in [38]. The main idea of a CMR is, in general, to represent using a single conditional rule more than one conventional table rules. This way the size of the transition function structure can be reduced substantially.

For the purposes of this paper, a conditional rule (CMR) for a 2D CA working with 5-cell neighbourhood is defined as $(cnd_N\ s_N)\ (cnd_W\ s_W)\ (cnd_C\ s_C)\ (cnd_E\ s_E)\ (cnd_S\ s_S) \rightarrow s_C^{new}$, where cnd_x denote a condition operator and s_x represent a state value. Each CMR thus consists of pairs: *(a condition and a state value)*, that corresponds to (is evaluated with respect to) a specific cell in cellular neighbourhood, and a new state value. For the experiments presented in this paper, the following conditions are considered: $= 0$, $\neq 0$, $\leq s$, $\geq s$, where s denotes an arbitrary state value from range 0 to 255. This means that in case of conditions $=$ and $\neq$ the cell states are always evaluated against state 0 and conditions $\leq$ and $\geq$ allows evaluating against any given state value. This set of conditions resulted from our long-term experimentation with cellular automata and its suitability was proven in several studies (e.g. recently [37], [38], [39]). A finite sequence of CMRs represents a transition function for a CA.

For example, consider a conditional rule $(\neq 0)(\geq 2)(= 0)(\geq 2)(\geq 1) \rightarrow 2$ and let c_N, c_W, c_C, c_E, c_S be cells in states $2, 3, 0, 3, 3$ respectively. The new state of c_C will be determined by evaluating the CMR-based transition function as follows. The CMRs are evaluated sequentially until a rule is found whose all conditions are true with respect to the cell states in the given neighbourhood. Consider the evaluation of the aforementioned conditional rule according to which $c_N \neq 0$ is true because $c_N = 2$, $c_W \geq 2$ is also true since $3 \geq 2$, $c_C = 0$ which satisfies the third condition and the last two conditions hold too because both c_E and c_S possess state 3. All conditions of the CMR were evaluated as true, therefore this CMR is said to match and will be applied to determine the next state of c_C (i.e. $s_C^{new} = 2$ according to the value on the right side of the CMR). If no matching rule in the sequence is found, then the cell keeps its current state. Note that the CMR-based transition functions can be transformed to the appropriate table rules which preserves the original concept of CAs as described in [37]. However, for the CA working with 256 cell states, that we deal in this paper, the table representation is intractable because of enormous memory requirements as mentioned above.

B. Evolutionary design of CMR-based cellular automata for image filtering

In this section the representation, fitness function and evolutionary algorithm will be described.

A sequence of CMRs is represented by an array of integers of a fixed size. Since 5-cell neighbourhood is considered, each CMR consists of 5 pairs of integers (one integer encodes the condition and the other the state value) and a single integer encoding the new state value. Hence in total a single CMR is composed of $5 \times 2 + 1 = 11$ integers. Note that for the set of conditions described in Section II-A, the following integers are considered for their encoding: 0 for $=$, 1 for $\neq$, 2 for $\leq$ and 3 for $\geq$. Let G denote the number of CMRs in a sequence of CMRs. Then each chromosome, encoding a transition function, consists of $G \times (5 \times 2 + 1)$ integers. The goal of the evolution is to find such a transition function according to which the CA behaviour satisfies (or optimises) given criteria.

For the purposes of image filter design, where I_{fil} denotes the corrupted image (i.e. the image to be filtered) and I_{ref} is its reference (uncorrupted) version, both of size $K \times L$ pixels, *mean square error* (MSE) was chosen as the fitness metric. The aim of the evolution is to minimize this criterion, i.e. the lower the MSE, the better the filter. MSE can be expressed as follows:

$$MSE = \frac{1}{KL} \sum_{i=1}^{K} \sum_{j=1}^{L} (I_{fil}[i,j] - I_{ref}[i,j])^2. \qquad (1)$$

In order to evaluate a candidate solution, represented by a CMR-based transition function for a CA, the $K \times L$-cell CA is initialised by the values of pixels of I_{fil}. Then three[1] steps of the CA are performed using the CMRs encoded in the chromosome. The current cell states are evaluated with

[1]Note this value was determined experimentally and is primarily based on the fact that only 5 pixels enter the filter, only a single transition function is considered, hence some intermediate steps may be required in order to eliminate (a substantial part of) the noise from the filtered image; this fully corresponds to the basic concept of uniform CA which allows us to more deeply investigate advanced techniques in the future.

respect to I_{ref} according to (1). The MSE is assigned to the chromosome as fitness value.

For the evolution of CMR-based transition functions, we applied a variant of (μ, λ)-Evolution Strategy algorithm, where μ is the size of parent population and λ is the number of offspring generated from the parent population ($\lambda > \mu$). The evolution works as follows. The initial parent population is generated randomly. The selection for reproduction, introduced for the purposes of this paper, is performed by a "round robin" scheme as follows. The parent chromosomes are selected deterministically one by one, each of which undergoes mutation of j integers, where j is the index of the parent in the population (from 1 to μ). The mutation is performed by randomly selecting j integers of the chromosome and replacing them by new valid randomly generated values. Since $\lambda > \mu$, after selecting the last parent the selection continues again from the first parent (round robin) until λ offspring are created. Then the offspring are evaluated by the fitness, sorted from the best to worst according to their fitness values and the best μ offspring replaces the original parent population. Moreover, the best-so-far solution is recorded during the evolution and replaced in case of any better one is detected. Note that the mutation of j integers, based on the index j of the parent in the (sorted) population, ensures that the best individual undergoes the smallest modification whilst in case of worse individuals more changes are allowed. The evolution repeats until a maximum number of generations are performed, then the best-so-far individual is returned as a result.

III. Experimental results

In this paper we consider the evolutionary design of image filters for filtering Salt-and-Pepper noise as a case study. Note that Salt-and-Pepper noise causes corruption of some pixels of the image by the maximal or minimal value of the pixel value scale. In particular, in case of 8-bit grayscale images, each corrupted pixel possesses either value 0 (i.e. black) or 255 (i.e. white). The intensity of the noise will be represented in percentages. For example, 10% noise of 200×200-pixel image means that 400 randomly selected pixels out of the total 40,000 pixels of the image are corrupted.

A. Objectives of experiments

The experiments conducted in this paper were focused on the following objectives. To demonstrate that the proposed variant of ES in combination with the CMR encoding allows an efficient evolution of high-quality filters with reduced resources of the CA. In particular we show that the evolution is able to provide in a short time image filters that can eliminate (or substantially reduce) Salt-and-Pepper noise of various intensities. Despite the fact that many currently known filters usually work with filtering windows of 3×3 or 5×5 pixels and the filtering algorithms are implemented as complex functions over 8-bit values (e.g. see [30]), in this paper we consider only 5-cell neighbourhood of the 2D CA as the inputs of the filter. Moreover, the training of the filter is performed using a single image which allows keeping a reasonable computational effort. As evident we provide significantly less resources to the design process and claim that even so the ES can provide solutions of high quality. The primary goal here is *not* to overcome the best filters known so far but to propose an alternative scheme (i.e. filtering by means of ordinary cellular automata) and demonstrate its usability.

B. Experimental setup

A wider analysis of ES-based evolution of cellular automata was published in [39] where we identified that the (μ, λ)-ES exhibits a good ability to avoid getting stuck in local optima in the task of designing complex CA. From this analysis we also identified a suitable ES setup for the experiments presented in this paper which is $\mu = 4, \lambda = 8$. For the image filter design, the maximum number of generations will be set to 100,000. Figure 1 shows the image utilised for training (i.e. fitness evaluation of candidate solutions). It is a picture of "Lena"[2] that is often considered for evaluating image processing experiments. The training image was corrupted by 10% noise. Several sets of experiments were conducted with various numbers of CMRs (in particular, we consider 5, 10, 20 and 30 CMRs the transition functions consist of). For each set of experiments 96 independent evolutionary runs were executed on the Salomon cluster[3] equipped by 2 x Intel Xeon E5-2680v3, 2.5 GHz, 12 cores per each computational node.

(a) corrupted by 10% noise (b) reference image

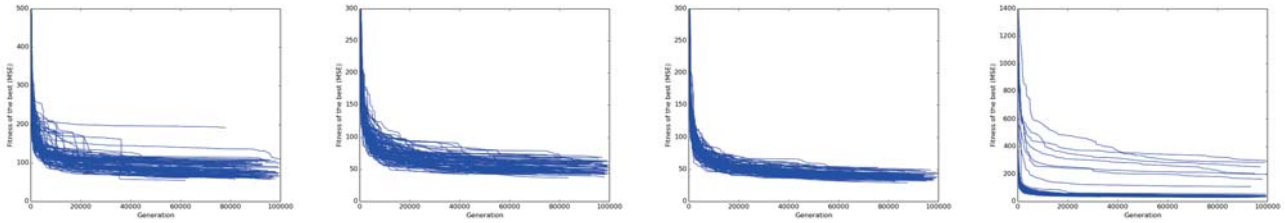
Fig. 1: Training image used for the evolution of image filters.

C. Statistics of evolutionary process

For each considered number of CMRs (5, 10, 20 and 30) 96 filters were obtained from each independent evolutionary run. In order to evaluate the performance of the evolutionary process, we observed the improvements of fitness of the best solution during each run. Figure 3 shows the progress for each run in each set of experiments considering the maximum 100,000 generations. As evident, in most cases there is a rapid improvement of the best fitness during the first 2000 generations and further improvements can be achieved during the rest of the generations for some runs. Therefore, if a filter

[2]https://kasunkosala.wordpress.com/computer-vision-and-digital-image-processing/

[3]see https://docs.it4i.cz/salomon/hardware-overview/



(a) experiments with 5 CMRs (b) experiments with 10 CMRs (c) experiments with 20 CMRs (d) experiments with 30 CMRs

Fig. 3: Progress of improving the best solution during the evolution of independent runs. Note that at the beginngin the fitness was around 6000 but the scale of the limit of the vertical axis was set lower due to illustration purposes. As can be seen, a rapid improvements can be observed in most runs which allows designing (prototype) filters in a couple of minutes).

```
!= 187 == 247 == 161 != 112 == 235 --> 178
==  53 >=  71 == 154 <= 254 <=  49 --> 133
!=  86 >=  68 ==  14 <= 255 >= 107 --> 255
>= 120 !=  85 <=  18 >=  90 >= 113 --> 143
!=  55 !=  49 <=  10 >=  59 >=  77 --> 96
== 203 <=  30 ==  73 >= 198 <=  10 --> 119
>=  43 !=  71 ==  31 >=  74 == 130 --> 247
== 137 >= 246 >=  81 == 207 !=  44 --> 167
!=  30 <=  98 >= 235 >=   1 <= 175 --> 6
>=  52 != 121 <=   6 !=  83 != 252 --> 71
>= 185 !=  49 >= 240 >= 182 != 124 --> 207
>= 222 != 191 <=   4 <= 157 >= 123 --> 0
<= 115 != 105 >= 248 <= 148 != 124 --> 4
>=   9 != 242 <=   6 != 104 != 142 --> 31
!= 109 >=  67 >= 242 != 212 >=  48 --> 160
<=  52 != 196 ==  39 <= 166 !=  48 --> 2
!= 134 >= 158 ==  83 <=  62 ==  60 --> 250
>=  97 != 248 == 158 ==   5 == 196 --> 199
!= 159 == 109 == 244 !=  19 !=  36 --> 254
<= 152 <= 159 >= 160 <= 145 <= 166 --> 133
```

Fig. 2: The evolved CMR representation of the F89 filter

needed to be designed very quickly, it would be possible to shorten the evolution time substantially by reducing the number of generations. We measured that such runs can be performed on a common PC or modern laptop in order to obtain a filter of average quality within several tens of seconds in average (or up to a few minutes for larger numbers of CMRs). This is particularly true for experiments with 10 and 20 CMRs in which we identified the best filters out of all experiments. This aspect represents one of the contributions of this paper – a possibility to obtain filters of a reasonable quality in very short time. Note that in [30] the average computation time on Xeon X5670, 2.93 GHz using an optimised CGP implementation and 250,000 generations was about 4 hours.

D. Evaluation of evolved filters

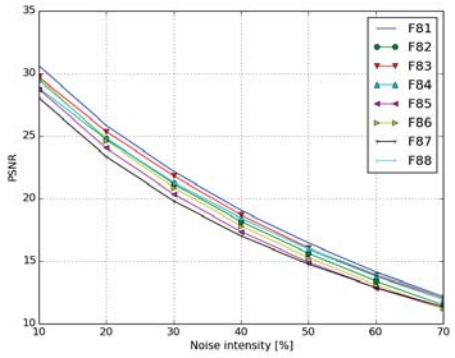
In order to evaluate the evolved filters, the Peak Signal-to-Noise Ratio (PSNR) was calculated for various images and noise intensities according to (2) which represents a common metric for the evaluation of image quality (the higher the PSNR, the better the filter and image quality).

$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{KL} \sum_{i=1}^{K} \sum_{j=1}^{L} (I_{fil}[i,j] - I_{ref}[i,j])^2} \quad (2)$$

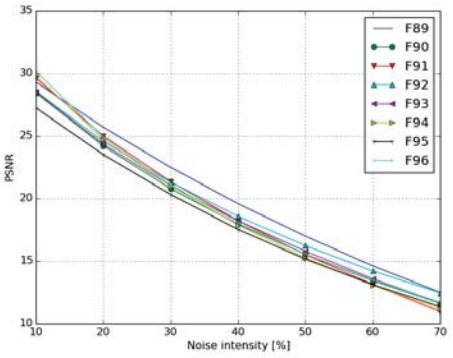
After training the filters during evolution we used a set of 24 test images from Vašíček's database[4] corrupted by the noise from 10% to 70% for the evaluation of obtained results. Figure 4 shows analysis of filters obtained from selected evolutionary runs in each set of experiments. The PSNRs exhibit similar (but not exactly the same) values for all sets of experiments in most cases, i.e. we obtained many various filters of similar (average) quality. As may be seen in Fig. 4c and 4d, for 30 CMRs in chromosomes there are several results (specifically F49, F51, F73, F74) that do not fit the usual PSNR course. In these cases, the evolution got stuck and was not able to further improve a suboptimal solution. Such filters do not work as expected and usually destroy the filtered image. The best filters achieved PSNR about value 30 for 10% noise which decreases to 12–15 in most cases for the highest noise intensity considered (70%).

In order to compare the results obtained in this paper with other solutions (both conventionally designed and evolved), the appropriate PSNRs are presented in Fig. 4e and 4f [30]. A significant difference of these solutions from our filters is that they all work with filtering windows 3×3 or 5×5 pixels and use advanced operations which means that their implementations are more complex. On the other hand, the proposed method using CMRs requires only basic relational operators. Despite this fact, the average PSNRs of the the conventional (median) filters from [30] are comparable to (some of them are even worse than) the values of our solutions. This is particularly true for Fig. 4e, where a single-step filtering is considered. Although the best evolved filters in [30] exhibit better PSNRs (between 30 and 35), the difference against the proposed results is not very high (the better filtering quality is expectable because of more resources used for their implementation). The image quality of those filters may further be improved by applying multi-step approach (similar to the CA development) as shown in Fig. 4f. The observations obtained from this comparison indicate that similar filtering quality can be obtained using the simplest concept of 2D cellular automata and with less computational effort which was the objective of this paper.

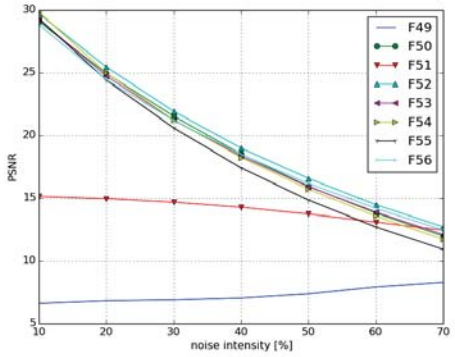
[4]https://www.fit.vutbr.cz/~vasicek/imagedb/



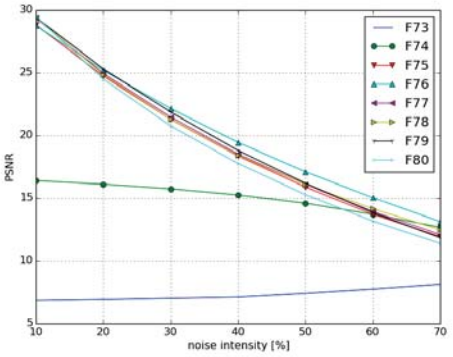
(a) runs 81–88 using 20 CMRs



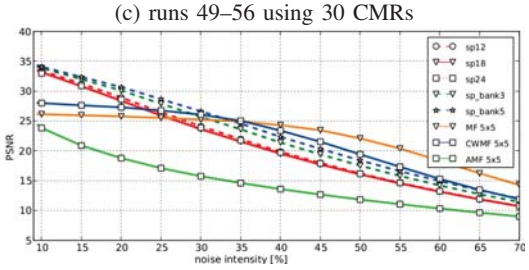
(b) runs 89–96 using 20 CMRs



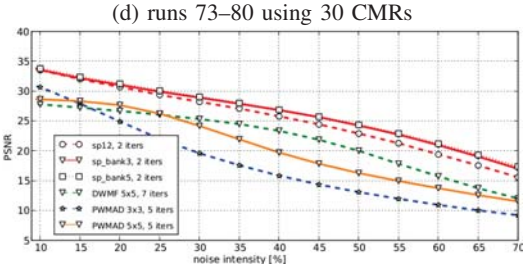
(c) runs 49–56 using 30 CMRs



(d) runs 73–80 using 30 CMRs



(e) single-step filtering from [30]



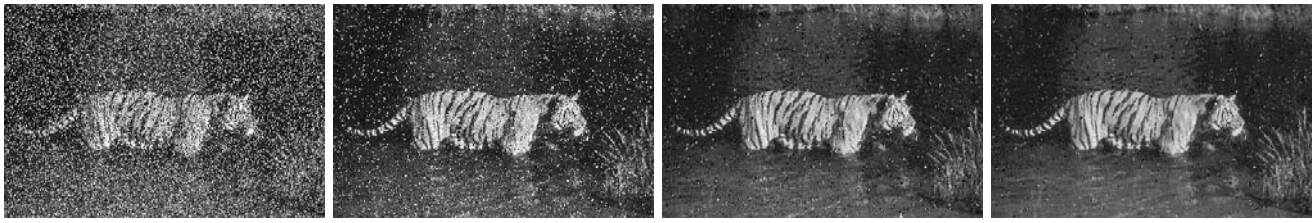
(f) multi-step filtering from [30]

Fig. 4: Analysis of filters from selected evolutionary runs. The filtering quality is expressed as average PSNR values evaluated for each filter and noise intensity from 10% to 70% using 24 test images from Vašíček's database (https://www.fit.vutbr.cz/˜vasicek/imagedb/). In all cases (a-d) the PSNRs were evaluated after 3 CA steps. Figures (e, f) are extracted from [30] for comparison purposes – the "sp" filters were evolved, the rest is conventionally designed.

Although the PSNR allows us to quantify the filter quality, it can not substitute visual evaluation of filtered images. For this purpose we chose one of promising filters and used it for filtering several images. Specifically, filter F89 will be considered from Fig. 4b whose PSNR analysis indicates its ability to filter higher noise intensities. Its CMR representation discovered by the evolution is shown in Figure 2. Figure 5 demonstrates filtering of 10% noise of the training image in 3 CA steps. Since this image was used to evaluate candidate filters during evolution, it is no surprise that the filter provides very good output quality. Although some pixels evidently remained corrupted, their values are closer to those of the reference image. However, it is important to remember that the filter works with 5 input pixels only and the result was evaluated after 3 CA steps. We verified that similar output quality can be observed on other images corrupted by 10% noise. A more interesting evaluation of the proposed filter is shown in Fig. 6 where 30% noise is considered. Note that neither such high noise intensity nor this image was seen by the filter during evolution. As can be observed, the filtered image also exhibits a very reasonable quality with only a few remaining noisy pixels. Again, 3 CA steps were sufficient to achieve this result. In order to demonstrate good features of the filter, it is important to determine whether it can generalise

(a) corrupted image (b) 1st step (c) 2nd step (d) 3rd step

Fig. 5: Demonstration of filtering 10% noise from the training image by one of the best evolved filters in 3 CA steps.



(a) corrupted image (b) 1st step (c) 2nd step (d) 3rd step

Fig. 6: Demonstration of filtering 30% noise by one of the best evolved filters in 3 CA steps.

(i.e. is able to filter other images than that on which it was trained, possibly with higher noise intensity). For this purpose the "selfie" of the author of this paper corrupted by 50% noise was chosen that exhibits a significantly different shape than both the training image and that evaluated in Fig. 6. In this case the filtering process was performed in 5 steps and the result is shown in Fig. 7. As can be seen, it is very hard to retrieve some details from the corrupted image (Fig. 7a). Clear shapes start to appear already after 2nd step (Fig 7c) and the best result (not changing significantly with further CA steps) can be achieved after 4th and 5th step – see Fig 7e and 7f respectively. One may conclude that, given the filter setup and noise intensity, the filter exhibits a reasonable output quality[5].

IV. Conclusions

An evolutionary method was presented for the design of image filters using two-dimensional uniform cellular automata. A technique called Conditionally Matching Rules was applied to represent transition functions for cellular automata working with 256 cell states. This approach allowed reducing the length of chromosomes for the evolution substantially which was a need for such high number of states since the traditional table-based encoding would require enormous memory space. The problem of removing Salt-and-Pepper noise from 8-bit grayscale images was considered as a case study. A cellular automaton was directly initialised by the values of pixels of a corrupted image and the evolution was applied to design a suitable transition function that is able to eliminate the noise from the image during ordinary development of the cellular automaton. We showed that high-quality filters can be obtained using only 5-cell neighbourhood of the cellular automaton and the best results provide a good output quality which is comparable with several existing solutions that require more resources. Moreover, the proposed evolutionary method demonstrated a high performance allowing us to design filters in very short time on a common PC.

The proposed method represents the simplest approach of filtering images by means of cellular automata in which the cell states are directly represented by the values of pixels. Considering the obtained results, there may be a possibility to further improve the concept. For example, Moore's 3×3-cell cellular neighbourhood can be used, an advanced selection of particular (subset of) CMRs to determined the next state may be considered or various sequentially applied transition functions might be performed. Moreover, further research will also be focused on more complex noise scenarios (e.g. random noise or impulse burst noise).

[5]the uncorrupted version of this image can be viewed at https://www.fit.vut.cz/person/bidlom/

References

[1] E. R. Dougherty and J. T. Astola, Eds., *Nonlinear Filters for Image Processing*, ser. SPIE/IEEE Series on Imaging Science & Engineering. SPIE/IEEE, 1999.

[2] M. O. Ahmad and D. Sundararajan, "A fast algorithm for two-dimensional median filtering," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 1364–1374, 1987.

[3] Z. Vasicek and L. Sekanina, "Novel hardware implementation of adaptive median filters," in *Proc. of 2008 IEEE Design and Diagnostics of*

(a) corrupted image (b) 1st step (c) 2nd step (d) 3rd step (e) 4th step (f) 5th step

Fig. 7: Demonstration of filtering 50% noise by one of the best evolved filters in 5 CA steps.

Electronic Circuits and Systems Workshop. IEEE Computer Society, 2008, pp. 110–115.

[4] S. Ko and Y. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Transactions on Circuits and Systems*, vol. 15, pp. 984–993, 1991.

[5] H. Hwang and R. Haddad, "Adaptive median filters: new algorithms and results," *IEEE Transactions on Image Processing*, vol. 4, no. 4, pp. 499–502, April 1995.

[6] Y. Dong, "A New Directional Weighted Median Filter for Removal of Random-Valued Impulse Noise," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 193–196, 2007.

[7] V. Crnojevic, V. Senk, and Z. Trpovski, "Advanced impulse detection based on pixel-wise MAD," *IEEE Signal Processing Letters*, vol. 11, no. 7, pp. 589–592, 2004.

[8] P. Koivisto, H. Huttunen, and P. Kuosmanen, "Training-based optimization of soft morphological filters," *Journal of Electronic Imaging*, vol. 5, no. 3, pp. 300–322, 1996.

[9] P. Koivisto, J. Astola, V. Lukin, V. Melnik, and O. Tsymbal, "Removing Impulse Bursts from Images by Training-Based Filtering," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 3, pp. 223–237, 2003.

[10] M. Nikolova, "A variational approach to remove outliers and impulse noise," *J. Math. Imaging Vis.*, vol. 20, no. 1-2, pp. 99–120, 2004.

[11] J. Zhu, S. Wang, X. Wu, and K. F.-L. Chung, "A novel adaptive svr based filter asbf for image restoration," *Soft Comput.*, vol. 10, no. 8, pp. 665–672, 2006.

[12] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin Heidelberg: Springer-Verlag, 2015.

[13] J. D. Lohn and G. S. Hornby, "Evolvable hardware: Using evolutionary computation to design and optimize hardware systems," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 19–27, 2006.

[14] J. Dumoulin, J. Foster, J. Frenzel, and S. McGrew, "Special Purpose Image Convolution with Evolvable Hardware," in *Real-World Applications of Evolutionary Computing – Proc. of the 2nd Workshop on Evolutionary Computation in Image Analysis and Signal Processing EvoIASP'00*, ser. LNCS, vol. 1803. Springer-Verlag, 2000, pp. 1–11.

[15] S. Marshall, N. Harvey, and D. Greenhalgh, "Design of morphological filters using genetic algorithms," in *Tenth European Signal Processing Conference*. EURASIP, 2000, pp. 389–392.

[16] R. Porter, "Evolution on FPGAs for Feature Extraction," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2001.

[17] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[18] L. Sekanina, L. S. Harding, W. Banzhaf, and T. Kowaliw, *Image Processing and CGP*, ser. Natural Computing Series. Springer Verlag, 2011, pp. 181–215.

[19] L. Sekanina, "Image filter design with evolvable hardware," in *Applications of Evolutionary Computing*, ser. LNCS, vol. 2279. Springer Verlag, 2002, pp. 255–266.

[20] A. Burian and J. Takala, "Evolved Gate Arrays for Image Restoration," in *Proc. of 2004 Congress on Evolutionary Computing CEC'04*. IEEE Publ. Press, 2004, pp. 1185–1192.

[21] Y. Zhang, S. Smith, and A. Tyrrell, "Intrinsic Evolvable Hardware in Digital Filter Design," in *Applications of Evolutionary Computing*, ser. LNCS, vol. 3005. Quimbra, Portugal: Springer, 2004, pp. 389–398.

[22] Z. Vasicek and L. Sekanina, "Reducing the area on a chip using a bank of evolved filters," in *Evolvable Systems: From Biology to Hardware*, ser. LNCS, vol. 4684. Springer Verlag, 2007, pp. 222–232.

[23] T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," *Pattern Recognition Letters*, vol. 16, pp. 341–347, 1994.

[24] Z. Vasicek, M. Bidlo, L. Sekanina, and K. Glette, "Evolutionary design of efficient and robust switching image filters," in *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE Computer Society, 2011, pp. 192–199.

[25] Z. Bao, F. Wang, X. Zhao, and T. Watanabe, "Fault-tolerant image filter design using ga," in *TENCON 2010 - 2010 IEEE Region 10 Conference*. IEEE, 2010, pp. 897–902.

[26] Z. Vasicek and L. Sekanina, "Hardware accelerator of cartesian genetic programming with multiple fitness units," *Computing and Informatics*, vol. 29, no. 7, pp. 1359–1371, 2010.

[27] J. Wang, Q. S. Chen, and C. H. Lee, "Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware," *IET computers and digital techniques*, vol. 2, no. 5, pp. 386–400, 2008.

[28] S. Harding, "Evolution of image filters on graphics processor units using cartesian genetic programming," in *2008 IEEE World Congress on Computational Intelligence*, J. Wang, Ed., IEEE Computational Intelligence Society. Hong Kong: IEEE Press, 1-6 Jun. 2008.

[29] Z. Vasicek and K. Slany, "Efficient phenotype evaluation in cartesian genetic programming," in *Proc. of the 15th European Conference on Genetic Programming*, ser. LNCS 7244. Springer, 2012, pp. 266–278.

[30] Z. Vašíček, M. Bidlo, and L. Sekanina, "Evolution of efficient real-time non-linear image filters for fpgas," *Soft Computing*, vol. 17, no. 11, pp. 2163–2180, 2013.

[31] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[32] P. Rosin, A. Adamatzky, and X. Sun, *Cellular Automata in Image Processing and Geometry*. Springer, 2014.

[33] P. L. Rosin, "Training cellular automata for image processing," *IEEE Trans. on Image Processing*, vol. 15, no. 7, pp. 2076–2087, July 2006.

[34] P. J. Selvapeter and Wim Hordijk, "Cellular automata for image noise filtering," in *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, Dec 2009, pp. 193–197.

[35] L. Diosan, A. Andreica, and A. Enescu, "The use of simple cellular automata in image processing," *Studia Universitatis Babe-Bolyai Informatica*, vol. 62, pp. 5–14, 06 2017.

[36] P. Kundra, H. M. Singh, V. Kumar, and P. Juneja, "Digital image edge detection using cellular automata," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 7S2, pp. 332–337, 2019.

[37] M. Bidlo, "On routine evolution of complex cellular automata," *IEEE Trans. on Evolutionary Computation*, vol. 20, no. 5, pp. 742–754, 2016.

[38] ——, *Advances in the Evolution of Complex Cellular Automata*, ser. International Joint Conference, IJCCI 2016 Porto, Portugal, November 9-11, 2016 Revised Selected Papers. Springer International Publishing, 2019, pp. 123–146.

[39] ——, "Comparison of evolutionary development of cellular automata using various representations," *MENDEL Soft Computing Journal*, vol. 2019, no. 1, pp. 95–102, 2019.