On Routine Evolution of New Replicating Structures in Cellular Automata

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

Keywords: Genetic Algorithm, Cellular Automaton, Transition Function, Conditional Rule, Replicating Loop.

Abstract: This paper presents evolutionary design of two-dimensional, uniform cellular automata. The problem of replicating loops is considered as a case study. Conditionally matching rules are used as a technique that is suitable to the design of cellular automata state transition rules. A genetic algorithm is applied to the design of cellular automata that satisfy the requirements of replicating loops. It is shown that such evolution is able to find various state transition rules that support replication of a given loop. Results presented herein demonstrate the ability of derived cellular automata to perform replication not only from an initial instance of the loop but also, that from a seed the loop can autonomously grow.

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. 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 spontaneous 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. Conditionally Matching Rules (CMR) are a representation of the transition rules, and were introduced, together with some early results related to binary CA, in (Bidlo and Vasicek, 2013) and (Bidlo, 2014).

Whilst the most of the aforementioned studies considered binary CA (i.e. those working with two cell states only), that 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 the information thanks to the ability of the 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 paper is to demonstrate an ability of the CMR approach to automatically design transition rules for CA that support replication of given structures in uniform, multi-state 2D array. A genetic algorithm will be applied in order to discover suitable transition rules that perform replication of the given loop-like 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.

2 FUNDAMENTALS OF CELLULAR AUTOMATA

The original concept of cellular automaton introduced in (von Neumann, 1966), that will be considered in this paper, 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 paper 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. In this paper, 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 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 paper, 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 paper. 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.

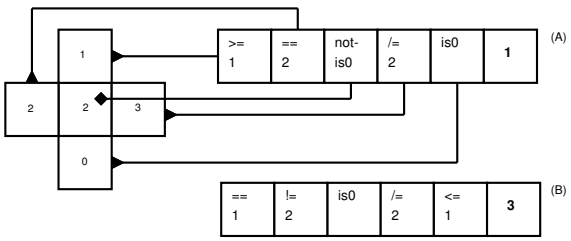

Figure 1: Example of a conditionally matching rule specified for 5-cell neighbourhood. The value of the new state is written in bold. (A) example of a matching CMR, (B) example of a CMR that does not match – the second and third condition is evaluated as false.

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 calculating the new state is deterministic (it is assumed that the condition functions are deterministic too). Therefore, it is possible to convert the CMR-based tran-

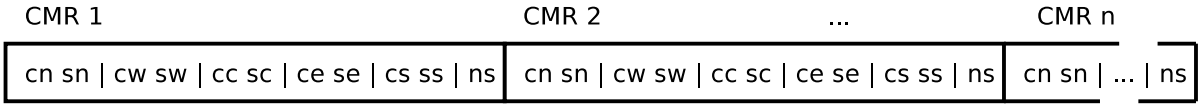

Figure 2: Structure of a chromosome for genetic algorithm encoding a CMR-based transition function. *cx* denote a condition for the cell at position *x* in the neighbourhood, *sx* represents the state value to be investigated using the appropriate condition with respect to the state of cell at position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

sition 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 paper.

For each experiment, the GA is executed for 4 millions of 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 performed evolutionary experiments and presents some results together with a more detailed analysis. Two sets of experiments are considered in which the CA works with 8 and 10 cell states. 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 (4 millions of generations) is approximately 12 hours.

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

A replicating loop is considered whose structure consists of 6 different non-zero states as shown in Figure 3a. The genetic algorithm is applied to the design of a transition function for the CA that performs 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 considered in order to evaluate the candidate solutions during evolution and calculate 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 required to be able to replicate again and again during the subsequent CA development. 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 obtained solutions are further investigated using a visual software simulator developed by the author of this paper 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 paper the term "general" means an ability of a solution to repeatedly produce more replicas of the given loop, not an ability to replicate arbitrary loops.

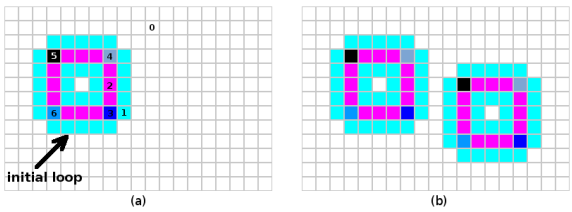

Figure 3: Structure of the replicating loop and cellular automaton of the size that was evaluated in the evolutionary experiments: (a) the initial CA state containing the loop to be replicated, (b) the target state specifying the replica arrangement.

Table 1 summarises results of the evolutionary experiments 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. More research is needed in order to optimise the evolutionary algorithm for this class of problems.

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 paper 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 in 16 steps (the best solution of this paper) 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 solutions discovered in this paper 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 required in the fitness function. Hence it can be considered as an additional feature of the evolved solution. However, it is not a common behaviour because only a few of the obtained transition functions are capable to do it.

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 4 millions of 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

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 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-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.

In order to perform a general evaluation of the obtained results within to 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 paper, 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 only the given loop 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 during evolution.

Although the shape of the proposed loop was inspired by the existing (self-replicating) loops and the GA provided some successful results to replicate this loop, 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 that requires a higher number of steps to finish the replica (e.g. Langton's loop needs 151 steps) which makes the evaluation of such solution very time-consuming (it is a case of the problem of scale during the fitness evaluation). 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 algorithms exist in the solution space for such loop, then the GA may not be able to find the solution in a reasonable time. Despite this issue, the obtained results bring some open questions whose investigation could be beneficial for the self-replication as well as 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 research.

6 CONCLUSIONS

In this paper the evolutionary discovery of new replication processes was proposed. It was shown that conditionally matching rules are suitable for a routine evolutionary design of multi-state CA that perform replication of a given loop-like structure. The experiments provided many different solutions how the replication of an initial loop can be performed. In addition, some of the transition functions demonstrated that the loop can even autonomously grow from a single-cell seed and subsequently replicate according to the original specification. It indicates that CA may exhibit some features that has not yet been known and has not been discovered so far using conventional techniques. A disadvantage of the proposed results may be seen in a low replication speed in comparison with some known replicating loops (the solutions presented herein replicate the loop in one direction only). However, optimization of the replication speed was not a goal of this paper. In general, it was demonstrated that new techniques of replication can be discovered automatically for a given loop-like structure.

ACKNOWLEDGEMENTS

This work was supported by the Czech Science Foundation via the project no. GA14-04197S *Advanced Methods for Evolutionary Design of Complex Digital Circuits*, and the *IT4Innovations Centre of Excellence*, no. 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.

REFERENCES

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

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

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

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

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

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

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

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.

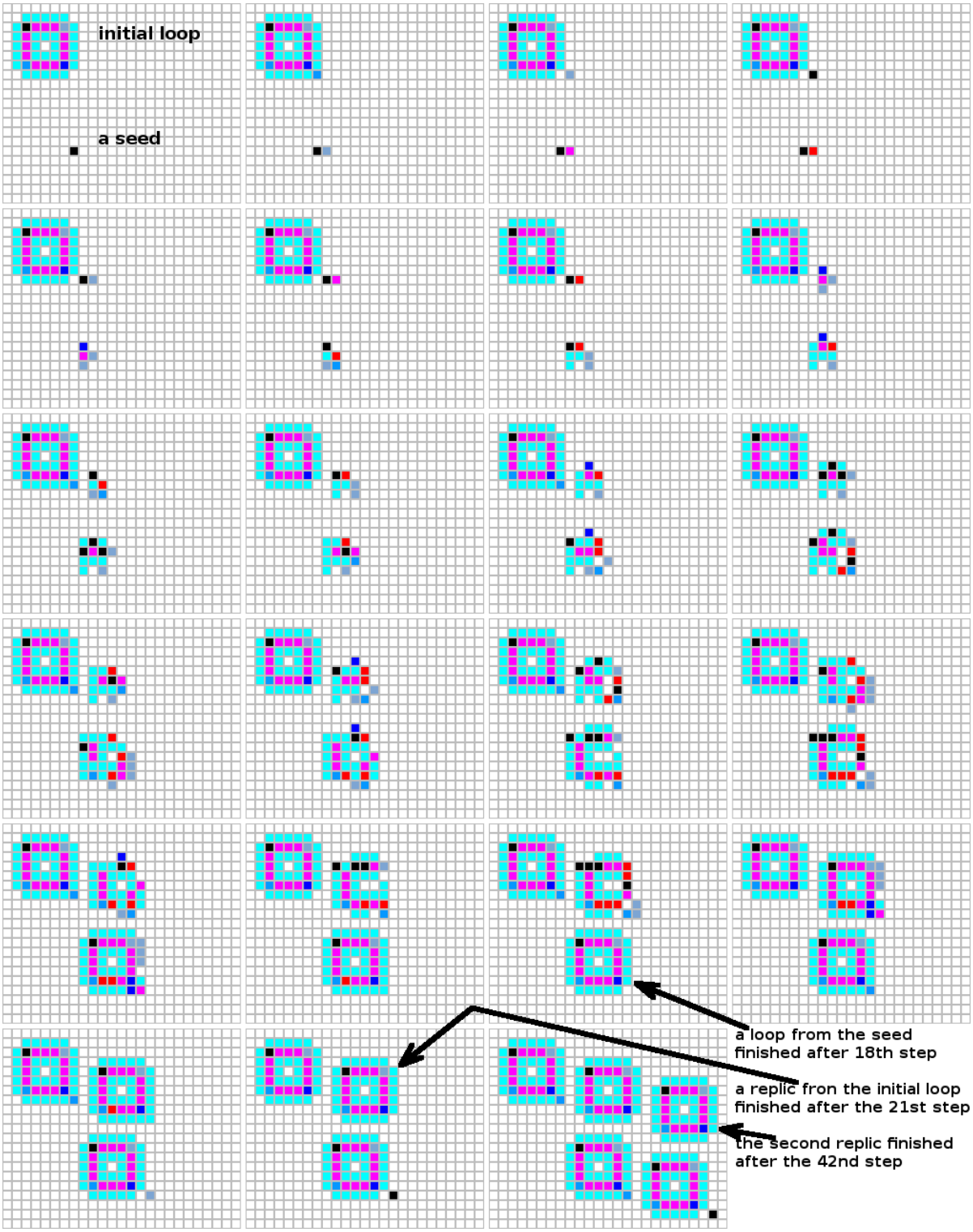

Figure 4: Develpment of a CA performing replication of the loop from Figure 3. The sequence of steps reads from left to right and top to bottom. The upper part of each step of the CA illustrates the replication of the initial loop. The bottom part demonstrates a seed represented by a cell in state 5. 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).

>=6;>=1;>=3;==0;>=1|7	==0;>=4;<=0;==0;!=0|1
>=0;<=3;>=0;<=0;>=3|1	<=2;==0;==0;<=2;>=1|3
==0;==0;==0;>=3;!=0|6	>=7;==0;>=6;>=6;<=2|2
<=1;!=0;<=1;>=5;<=1|0	<=5;!=0;>=7;!=0;<=0|7
==0;!=0;>=3;<=2;!=0|5	>=0;==0;==0;!=0;!=0|1
>=4;>=0;>=7;==0;>=2|6	!=0;!=0;==0;==0;==0|3
<=7;>=5;!=0;<=4;!=0|2	<=1;<=0;!=0;!=0;>=6|7
<=5;<=3;==0;<=7;<=1|0	==0;>=4;<=2;<=7;>=6|1
>=3;>=3;!=0;>=0;!=0|6	>=5;>=1;>=1;==0;==0|0
!=0;>=4;<=4;>=5;==0|2	<=7;==0;==0;<=5;<=3|5
<=3;<=1;!=0;==0;<=2|1	!=0;>=5;==0;!=0;!=0|3
==0;>=5;<=3;==0;<=3|4	==0;==0;==0;>=5;!=0|6
==0;!=0;>=2;==0;!=0|4	==0;<=3;==0;==0;!=0|7
!=0;>=5;>=6;==0;<=6|4	!=0;==0;>=5;!=0;>=7|3
<=7;==0;>=0;<=2;!=0|1	!=0;<=6;>=5;<=2;<=0|7
>=1;>=7;!=0;==0;==0|7	==0;<=0;>=0;<=2;==0|0
<=7;>=3;<=6;<=3;>=3|4	==0;==0;==0;>=2;<=4|5
!=0;!=0;>=6;!=0;==0|6	==0;>=5;>=0;>=5;>=2|7
==0;==0;>=0;==0;<=3|5	==0;<=4;==0;>=6;<=2|6
<=1;<=3;>=3;>=7;!=0|1	==0;<=0;>=1;>=4;==0|2
>=4;<=3;!=0;==0;!=0|2	>=5;==0;==0;<=2;>=6|4
==0;<=0;==0;<=6;<=1|7	!=0;!=0;<=7;>=5;==0|3
>=0;>=5;<=7;>=2;==0|6	!=0;>=2;!=0;==0;==0|6
<=1;>=2;!=0;>=5;<=7|2	<=0;<=2;<=5;!=0;>=7|3
>=7;<=2;==0;<=6;<=1|1	>=6;==0;<=3;<=5;<=1|1

(a)

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

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

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

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

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

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

(b)

Figure 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 in this paper.

Figure 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.

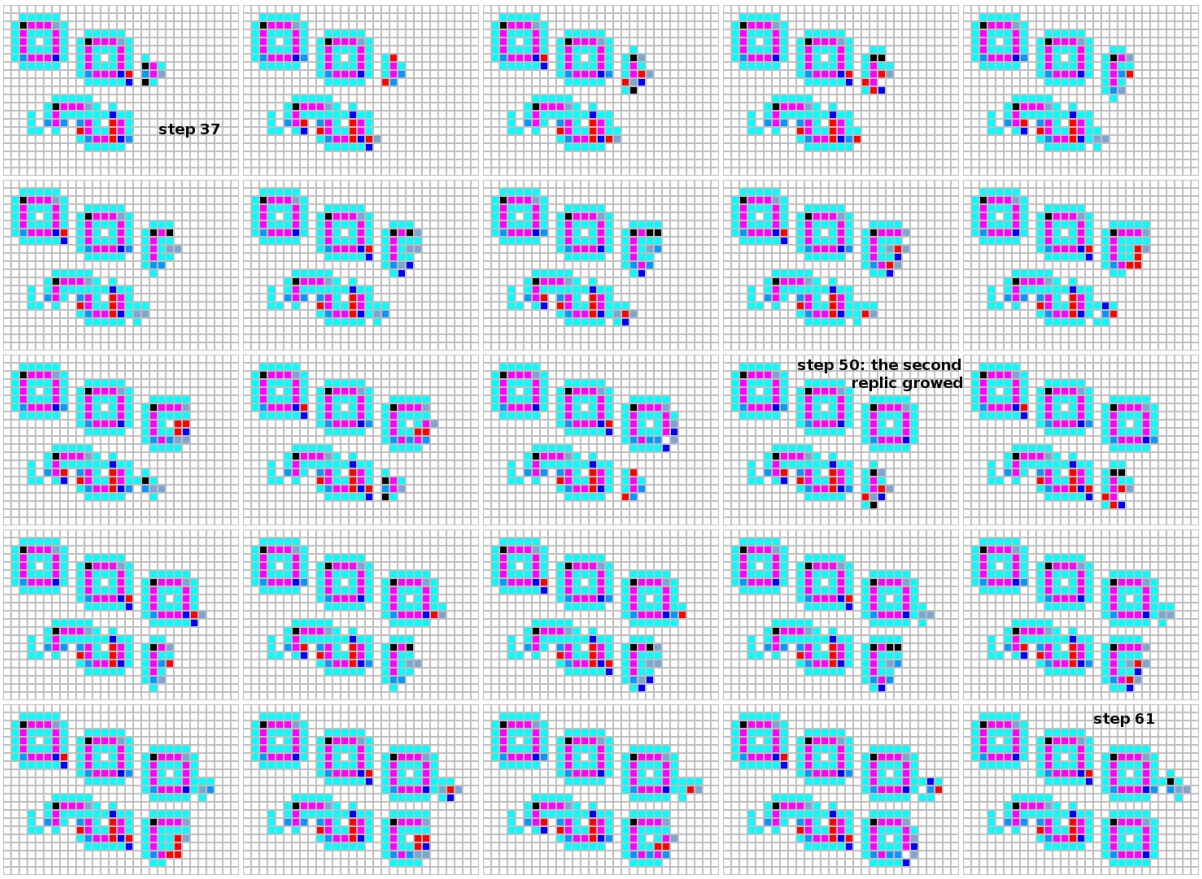

Figure 7: Part 2 of the replication according to the transition function from Figure 5. The sequence of steps reads from left to right and top to bottom. The development shows an autonomous growth of the loop from a non-specific structure that emerged in the last step of Figure 6 (the bottom part of 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.