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.

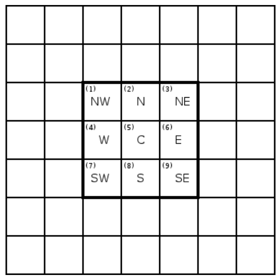


Fig. 1. Structure of Moore neighborhood in a cellular automaton. The neighborhood includes the investigated (central) cell C and its immediate neighbors. The numbers in parentheses denote the indices of cells in the neighborhood that will be considered in this paper.

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 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.

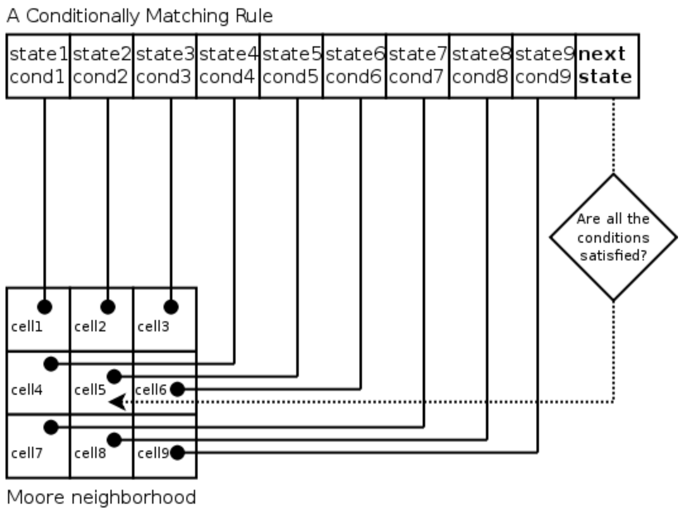


Fig. 2. Structure and interpretation of a conditionally matching rule

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

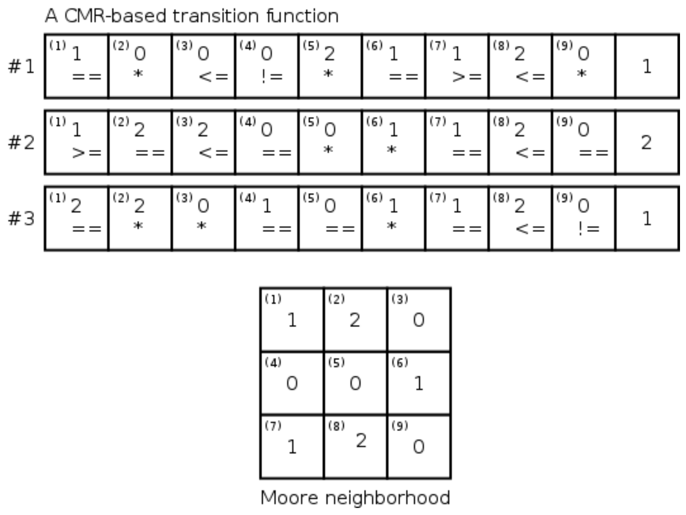


Fig. 3. Example of CMR-based transition function consisting of three conditionally matching rules

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(ca1, 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	**Success**	**Mean number**	**Number of table rules:**		
CMR	**rate**	**of gen. (std. dev.)**	**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

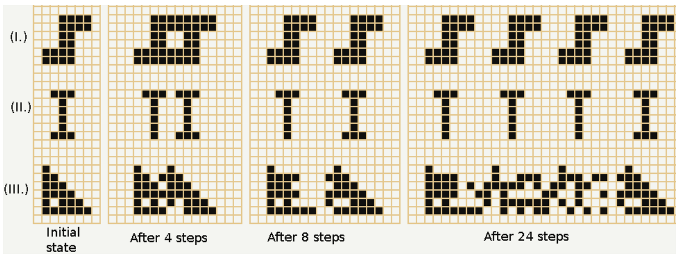


Fig. 5. Replication of some selected initial structures in a cellular automaton. This case represents an example of non-universal replicator, some structures are not replicated correctly.

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.

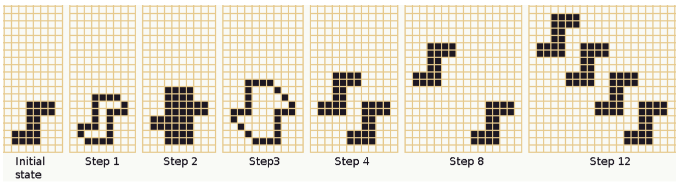


Fig. 6. Diagonal replication of an initial structures using the transition function from Figure 7

```
(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	**Success**	**Mean number**	**Number of table rules:**		
CMR	**rate**	**of gen. (std. dev.)**	**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.

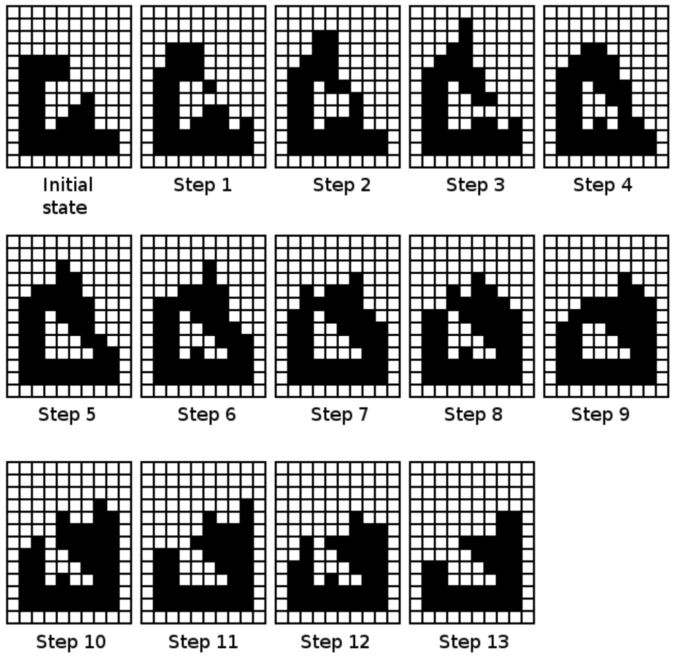


Fig. 8. Counter-clockwise rotation by 90 degrees in a cellular automaton. The pattern to be rotated is represented by the initial state. The rotation is performed in 13 steps using the transition function from Figure 9.

```
(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.

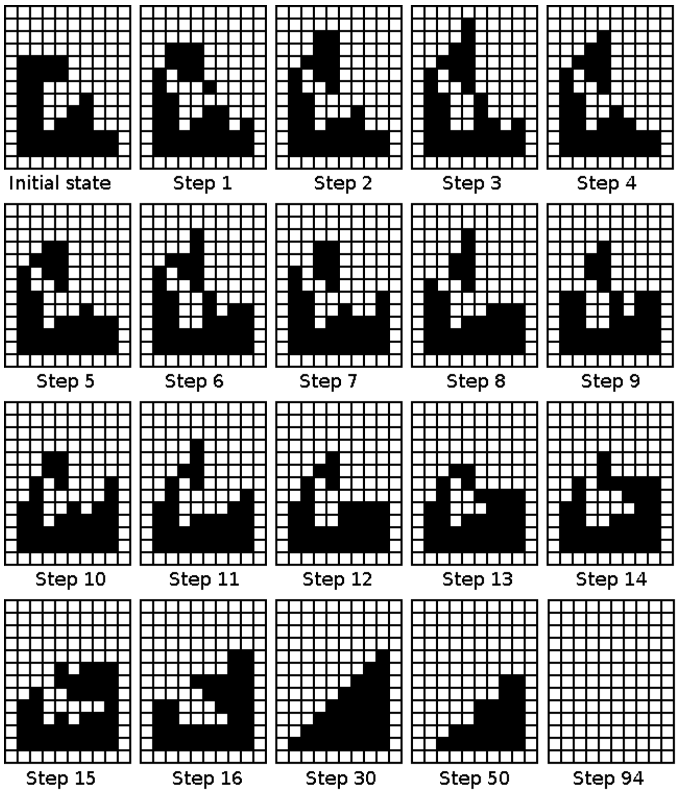
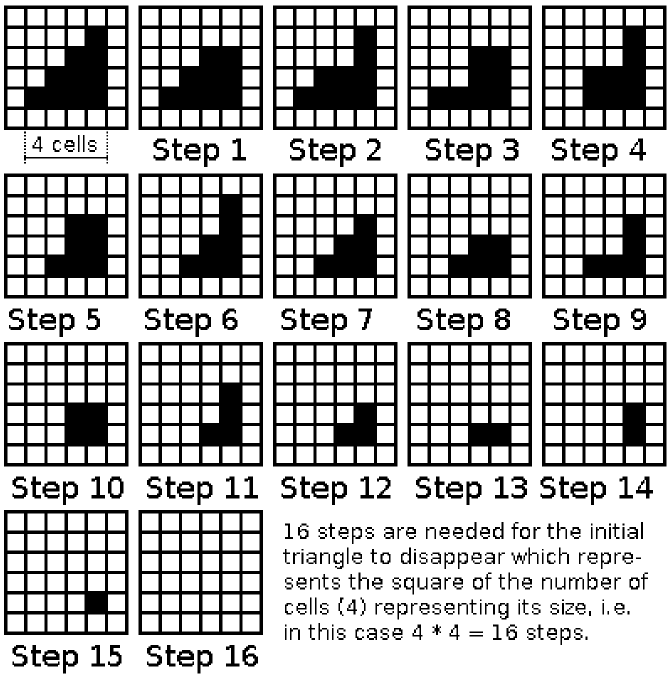


Fig. 10. Counter-clockwise rotation by 90 degrees in a cellular automaton according to the transition function from Figure 11

```
(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).



16 steps are needed for the initial triangle to disappear which represents the square of the number of cells (4) representing its size, i.e. in this case 4 * 4 = 16 steps.

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.