Millions of 5-State n^3 Sequence Generators via Local Mappings

Tien Thao Nguyen and Luidnel Maignan

Univ Paris Est Creteil, LACL, F-94010 Creteil, France

Abstract

In this paper, we come back on the notion of local simulation allowing to transform a cellular automaton into a closely related one with different local encoding of information. In a previous paper, we applied it to the Firing Squad Synchronization Problem. In this paper, we show that the approach is not tied to this problem by applying it to the class of Real-Time Sequence Generation problems. We improve in particular on the generation of n^3 sequence by using local mappings to obtain millions of 5-state solution, one of them using 58 transitions. It is based on the solution of Kamikawa and Umeo that uses 6 states and 74 transitions. Then, we explain in which sense even bigger classes of problems can be considered.

1 Introduction

1.1 Local Mappings to Explore the Cellular Solution Space

This paper is about the formal concepts of *local mapping* and *local simulation* which found their firsts applications in the study of the so-called Firing Squad Synchronization Problem (FSSP) proposed by John Myhill in 1957. In the latter, the goal is to find a single cellular automaton such that any one-dimensional horizontal array of an arbitrary number of cells *synchronizes*, *i.e.* such that a special state is set for all the cells at the same time. As explained in [8], there was a race to obtain solutions with as few states as possible, leading in 1987 to a situation where it was established, for minimal-time time solutions, that there were no 4-state solutions, and only one solution had 6-state using a unique strategy. In 2018, a surprise came when 718 solutions were found using massive computing power, but a bigger surprise came in 2020 when many millions of solutions where discover using ordinary computer power. It is still not known whether this is a 5 state solution, but the concept used to generate the millions of solutions, local mappings and local simulation still have many things to tell, a story that is not tied to FSSP, but that may lead to some ideas about the famous 5-state FSSP question. More information can be found in [9, 8].

1.2 Real-Time Sequence Generation Problems

To show explicitly in which sense the approach can be applied to other problems, let us focus here on the so-called Real-Time Sequence Generation problems, or

RTSG problems for short. In the latter, given a fixed sequence $S \subseteq \mathbb{N}$, the goal is to find a cellular automaton running on an one-dimensional horizontal array of cells such that the leftmost cell is in a special state exactly when the number of transition t from the beginning belongs to S. A formal description is given later. In the following, we write $f(n)$ to mean $S = \{f(n) \mid n \geq 1\}$.

The study of such problems began in 1965 for the sequence of prime numbers, with a description of a cellular automaton algorithm by Fischer [1]. In 1998, Korec [6] proposed a 9-state solution. Other sequences where considered in 2007 by Kamikawa and Umeo [7] who gave some different algorithms for the sequences 2^n, n^2, and 3^n using one-bit inter-cell-communication cellular automata. In 2012, Kamikawa and Umeo [3] described the sequence generation powers of CAs having a small number of states, focusing on the CAs with one (only one sequence n of all positive natural numbers), two, and three internal states, respectively. The authors enumerate all of the sequences generated by two-state CAs (linear sequences: $2n, 4n, 3n-1, n, 3n-2, 2n-1, n+1$; non-regular sequences: $2^{n+1}-2, 2^n-1$) and present several non-regular sequences like $2^n, n^2, 3^n$ that can be generated in real-time by three-state CAs, but not generated by any two-states CA. In 2016 [4], they gave a construction for the Fibonacci sequence using five-state, followed in 2019 [5] by two solutions of 8 states and 6 states for the sequence n^3. In these studies, much attention has been paid to the developments of real-time generation algorithms and their small-state implementations on CAs for specific non-regular sequences. Other complexities are also studied such as the space, communication or state-change complexities.

Here we consider the sequence n^3 and provide millions of 5 states solutions using local mappings to improve on the previous 6-state solution. This reduction from 6-state to 5-state is reminiscent of the FSSP situation. For the particular n^3 sequence, this leaves open the question of the existence of solutions with 3 or 4 states. We did not try to improve on any other work listed above, the goal here being simply to illustrate the generality of the approach.

1.3 Organization of the Content

In Section 2, we begin by defining formally cellular automata, local mappings, local simulations, RTSG solutions and related objects in a suitable way for this study. In Section 3, we explain how local mappings can be used firstly to obtain a first 5-state solution to the n^3-RTSG problem, and then to generate millions of other solutions. These other solutions are essentially the same, but differ in the way the local information is encoded, leading to different numbers of transitions for example. This is in direct comparison with compiler optimization where a program is optimized but stays essentially the same. We finish this section by making more precise the generality of the approach. We conclude in Section 4 with a discussion of some additional aspects of this investigation, in particular with the relation with some topological concepts.

2 Preliminaries

We summarize here the formal definitions of cellular automata, local mappings solutions as defined in [8]. More detailed explanations can be found in [8, 9].

We then define RTSG solutions in a formally relevant way for this framework. As for the other papers, the formal setting is presented for one dimensional cellular automata with usual neighborhood $\{-1, 0, +1\}$, but is easily extended to any (non necessarily commutative) group and any (none necessarily fixed) neighborhood.

2.1 Cellular Automata, Local Mappings, and Local Simulations

The purpose of these following definitions is to describe cellular automata with partial transition table first directly and then in terms of their deterministic family of space-time diagrams. With this more explicit representation, the concepts of local mapping and local simulations are more easily understood. Non necessarily deterministic family of space-time diagrams also plays a role in this story.

Definition 1 *A* cellular automaton α *consists of a finite set of* states Σ_α*, a set of* initial configurations $\mathrm{I}_\alpha \subseteq {\Sigma_\alpha}^{\mathbb{Z}}$ *and a partial function* $\delta_\alpha : {\Sigma_\alpha}^3 \nrightarrow \Sigma_\alpha$ *called the* local transition function *or* local transition table*. The elements of* ${\Sigma_\alpha}^{\mathbb{Z}}$ *are called* (global) configurations *and those of* ${\Sigma_\alpha}^3$ *are called* local configurations*. For any* $c \in \mathrm{I}_\alpha$*, its* space-time diagram $\mathrm{D}_\alpha(c) : \mathbb{N} \times \mathbb{Z} \to \Sigma_\alpha$ *is defined as:*

$$\mathrm{D}_\alpha(c)(t,p) = \begin{cases} c(p) & \text{if } t = 0, \\ \delta_\alpha(c_{-1}, c_0, c_1) & \text{if } t > 0 \text{ with } c_i = \mathrm{D}_\alpha(c)(t-1, p+i). \end{cases}$$

The partial function δ_α *is required to be such that all space-time diagrams are totally defined. When* $\mathrm{D}_\alpha(c)(t,p) = s$*, we say that, for the cellular automaton* α *and initial configuration* c*, the cell at position* p *has state* s *at time* t*.*

Definition 2 *A* family of space-time diagrams D *consists of a set of states* Σ_D *and an arbitrary set* $D \subseteq {\Sigma_D}^{\mathbb{N}\times\mathbb{Z}}$ *of space-time diagrams. The* local transition relation $\delta_D \subseteq {\Sigma_D}^3 \times \Sigma_D$ of D *is defined as:*

$$((c^0_{-1}, c^0_0, c^0_1), c^1_0) \in \delta_D :\Leftrightarrow \exists (d,t,p) \in D \times \mathbb{N} \times \mathbb{Z} \text{ s.t. } c^j_i = d(t+j, p+i).$$

We call D *a* deterministic family *if its local transition relation is functional.*

Definition 3 *Given a deterministic family* D*, its* associated cellular automaton Γ_D *is defined as having the set of states* $\Sigma_{\Gamma_D} = \Sigma_D$*, the set of initial configurations* $\mathrm{I}_{\Gamma_D} = \{d(0,-) \in {\Sigma_\alpha}^{\mathbb{Z}} \mid d \in D\}$*, and the local transition function* $\delta_{\Gamma_D} = \delta_D$*.*

Definition 4 *Given a cellular automaton* α*, its* associated family of space-time diagrams *(abusively denoted)* D_α *is defined as having the set of states* $\Sigma_{\mathrm{D}_\alpha} = \Sigma_\alpha$*, and the set of space-time diagrams* $\{\, \mathrm{D}_\alpha(c) \mid c \in \mathrm{I}_\alpha \,\}$ *and is clearly deterministic.*

Definition 5 *A* local mapping ℓ from a CA α to a finite set X *consists of two functions* $\ell_{\mathtt{z}} : \{d(0,p) \mid (d,p) \in \mathrm{D}_\alpha \times \mathbb{Z}\} \to X$ *and* $\ell_{\mathtt{s}} : dom(\delta_\alpha) \to X$*. We define its* associated family of diagrams $\Phi_\ell = \{\ell(d) \mid d \in \mathrm{D}_\alpha\}$ *where:*

$$\ell(d)(t,p) = \begin{cases} \ell_{\mathtt{z}}(d(0,p)) & \text{if } t = 0, \\ \ell_{\mathtt{s}}(d(t-1,p-1), d(t-1,p), d(t-1,p+1)) & \text{if } t > 0. \end{cases}$$

If Φ_ℓ *is deterministic, we say that* ℓ *is a* local simulation from CA α to CA Γ_{Φ_ℓ}*.*

Q	Right State					
Left State	Q	A	B	C	D	E
Q	Q	Q	B	C	D	D
A						
B	C					
C	Q					
D	Q					
E	Q					
⋆	Q	Q	B	E	E	E

A	Right State					
Left State	Q	A	B	C	D	E
Q			Q	Q	Q	Q
A						
B						
C						
D						
E						
⋆			Q	Q		

B	Right State					
Left State	Q	A	B	C	D	E
Q	B		B			
A			C	C		
B			B	B		
C			B	B		
D						
E						
⋆	A		A			

C	Right State					
Left State	Q	A	B	C	D	E
Q			C	C	C	C
A	E		A	A	A	A
B	C					
C	C		C	C	C	C
D						
E			A	A	A	A
⋆						

D	Right State					
Left State	Q	A	B	C	D	E
Q	D				D	D
A	B				C	C
B						
C	D				D	D
D	D				D	D
E	B				C	C
⋆						

E	Right State					
Left State	Q	A	B	C	D	E
Q	E					
A	D					
B						
C	E					
D	E					
E	D					
⋆				Q	Q	Q

Figure 1: Transition table of Kamikawa and Umeo's 6-state solution using 74 transitions.

2.2 Real-Time Sequence Generators

Definition 6 *A cellular automaton is* RTSG-candidate *if there are four special states* $\star_\alpha, \mathtt{B}_\alpha, \mathtt{Q}_\alpha, \mathtt{S}_\alpha \in \Sigma_\alpha$*, if* $\mathrm{I}_\alpha = \{\overrightarrow{\infty}_\alpha\}$ *with* $\overrightarrow{\infty}_\alpha$ *being the* RTSG initial configuration *infinite to the right, i.e.* $\overrightarrow{\infty}_\alpha(p) = \star_\alpha$*,* $\mathtt{B}_\alpha$ *and* $\mathtt{Q}_\alpha$ *if* p *is respectively* $p \leq 0$*,* $p = 1$ *and* $p \geq 2$*. Moreover,* $\star_\alpha$ *must be the* outside state, i.e. *for any* $(c_{-1}, c_0, c_1) \in dom(\delta_\alpha)$*, we must have* $\delta(c_{-1}, c_0, c_1) = \star_\alpha$ *if and only if* $c_0 = \star_\alpha$*. Also,* $\mathtt{Q}_\alpha$ *must be a* quiescent state *so* $\delta_\alpha(\mathtt{Q}_\alpha, \mathtt{Q}_\alpha, \mathtt{Q}_\alpha) = \delta_\alpha(\star_\alpha, \mathtt{Q}_\alpha, \mathtt{Q}_\alpha) = \mathtt{Q}_\alpha$*.*

The $\star_\alpha$ state is not really counted as a state since it represents cells that should be considered as non-existing. Therefore, an RTSP-candidate cellular automaton α will be said to have s states when $\mid \Sigma_\alpha \setminus \{\star_\alpha\} \mid = s$, and m transitions when $\mid \mathrm{dom}(\delta_\alpha) \setminus \Sigma_\alpha \times \{\star_\alpha\} \times \Sigma_\alpha \mid = m$.

Definition 7 *Given a sequence* $S \subseteq \mathbb{N}$*, a RTSG-candidate cellular automaton* α *is a* S-RTSG solution *if for any time* t*,* $\mathrm{D}_\alpha(\overrightarrow{\infty}_\alpha)(t, 0) = \mathtt{S}_\alpha$ *if and only if* $t \in S$*.*

Proposition 8 *There is a* n^3*-RTSG solution using 6 states and 74 transitions.*

Proof

In Figure 1 is the solution of Kamikawa and Umeo, reproduced with the same format as their paper to ease comparison. Also, the local transition function δ contains the above entries and additional obvious entries for the outside state $\star$. The proof of correction can be found in [5].

The space-time diagram of this solution is depicted up in the two left columns of Figure 2, where the cell at position 0 has the state A at time 1, 8, 27, and

64 as expected. In [5], note that table of D wrongly has column C filled with the content of column E. This mistake is easy to catch by examining the proofs and space-time diagrams of the paper.

3 Exploring RTSG Solutions and More via Local Mappings

Let us now describe how to apply the same techniques used for the FSSP in [8] in order to first obtain a first optimization from 6-state to 5-state, and then use the exploration algorithm to generate millions of other 5-state solutions. The first step is to study those local mappings which *complies* with RTSG problems.

3.1 Compliant Local Mappings

Given two cellular automata α and β, a local mapping between them associates to each triplet found in a space-time diagram d at position p and time t of α to the state found in the associated diagram d' at position p and time $t+1$. When both of these cellular automata are RTSG solutions, this implies the following properties on the local mapping.

Definition 9 *A local mapping ℓ from an RTSG solution α to the states Σ_β of an RTSG-candidate CA β is said to be* RTSG-compliant *if it is such that (0) $\ell_{\mathtt{z}}$ maps $\star_\alpha$, $\mathtt{B}_\alpha$, and $\mathtt{Q}_\alpha$ respectively to $\star_\beta$, $\mathtt{B}_\beta$, and $\mathtt{Q}_\beta$, (1) $\ell_{\mathtt{s}}(c_{-1}, c_0, c_1) = \star_\beta$ if and only if $\delta_\alpha(c_{-1}, c_0, c_1) = \star_\alpha$ (meaning simply $c_0 = \star_\alpha$), (2) $\ell_{\mathtt{s}}(\star_\alpha, c_0, c_1) = \mathtt{S}_\beta$ if and only if $\delta_\alpha(\star_\alpha, c_0, c_1) = \mathtt{S}_\alpha$, and (3) $\ell_{\mathtt{s}}(\mathtt{Q}_\alpha, \mathtt{Q}_\alpha, \mathtt{Q}_\alpha) = \ell_{\mathtt{s}}(\star_\alpha, \mathtt{Q}_\alpha, \mathtt{Q}_\alpha) = \mathtt{Q}_\beta$.*

Proposition 10 *Given a sequence $S \subseteq \mathbb{N}$, let α be an S-RTSG solution CA, β an S-RTSG-candidate CA and ℓ a local simulation from α to β. β is an S-RTSG solution if and only if ℓ is RTSG-compliant.*

Proof

To see this, consider the diagram $d \in \mathrm{D}_\alpha$ of the solution α. The special RTSG states appear at specific places and ℓ ensures or witnesses, depending on the direction of the implication considered, that these special states/places are conserved in $\ell(d) \in \mathrm{D}_\beta$, (Definition 5). Indeed, condition (0) is just about the initial configuration, condition (1) is about the conservation of the outside state, condition (2) is about the conservation of the special generation state for the leftmost cell only and condition (3) about the conservation of the quiescent state behaviour. These conditions are sufficient to ensure and β is a solution, and clearly necessary since they perfectly match Definitions 6 and 7 of the problem.

Note that once α fixed, β can be reconstructed from ℓ, and ℓ from β. So the local mapping ℓ is just another representation of the RTSG-candidate β that

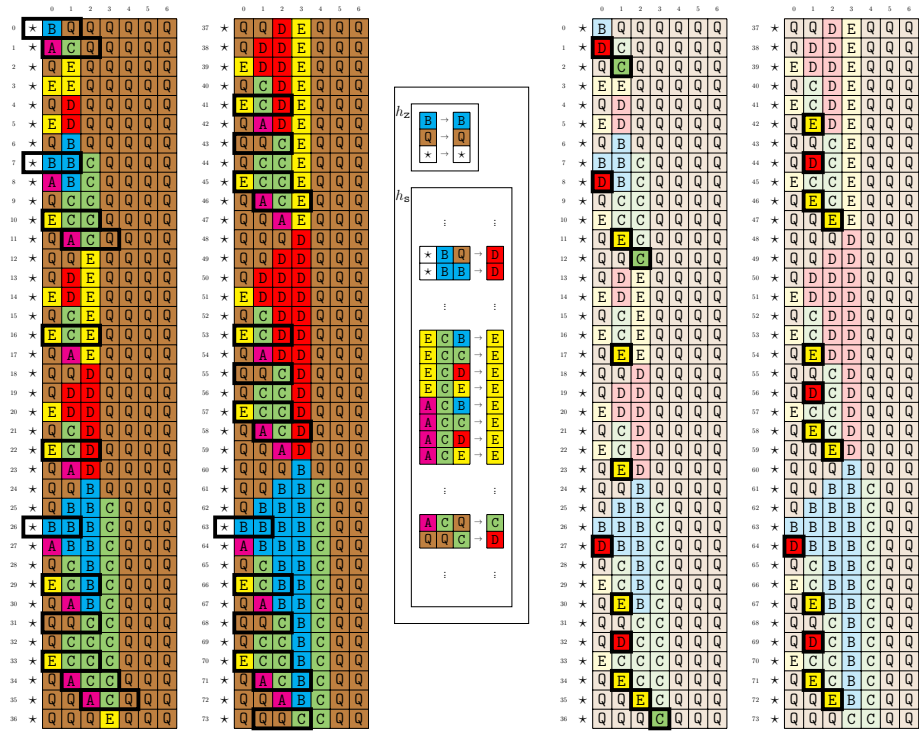

Figure 2: 6-state diagram, hand-crafted local mapping and resulting 5-state diagram.

it generates (see Definition 5), but it is much easier to check the compliance of the local mapping than the correction of CA β as an RTSG solution, and this is the key property than justifies this particular application of local mappings.

3.2 A Hand-Crafted Local Simulation

The first local mapping that we consider the *identity local mapping id* given by the local transition function of the 6-state solution of Proposition 8. This local mapping simply transforms this solution into itself. The point here is that we can now work with the local mapping. However, the reader should be careful to clearly distinguish modifications made the *id* local mapping, and the resulting modifications in the transition table. It is easier to think in terms of space-time diagram, since each modification in the local mapping corresponds directly to a uniform set of modifications in the space-time diagram which may or may not be deterministic after these modifications.

Let us now describe how the second, hand-crafted, local mapping is obtained, as we come back on the process itself later. It is build by noticing different features of the original space-time diagram on the left of Figure 2. The first thing is that the state A is not often used, so we can try to remove it entirely. This means changing every entry (x, y, z) of id_{s} such that $id_{\mathsf{s}}(x, y, z) = A$. But since A is the special generating state, we can not replace it by B, Q or E since they already appear in the evolution of the leftmost cell. So we can consider

Q	Right State				
Left State	Q	B	C	D	E
Q	Q	B	D	D	Q
B	C				
C	Q				
D	Q				
E	Q				
⋆	Q	B	E	E	Q

B	Right State				
Left State	Q	B	C	D	E
Q	B	B			
B		B	B		
C		B	B		
D		C	C		
E		C	C		
⋆	D	D			

C	Right State				
Left State	Q	B	C	D	E
Q	E	C	C	C	C
B	C				
C	C	C	C	C	C
D	C	C	C	C	C
E	C	E	E	E	E
⋆					

D	Right State				
Left State	Q	B	C	D	E
Q	D		C	D	D
B					
C	D			D	D
D	D			D	D
E	B			C	C
⋆		Q	Q		

E	Right State				
Left State	Q	B	C	D	E
Q		Q	Q	Q	Q
B					
C	E				
D	E				
E	D				
⋆			Q	Q	Q

Figure 3: Transition table of the hand-crafted 5-state solution using 72 transitions.

either C or D. However, looking at time 1, we see changing A into C would lead to a CCQ local configuration, which is already used. So we heuristically choose D instead, to have DCQ at time 1, an unused local configuration. To summarize, for the leftmost cell we choose to change A by D, and for the other cells, we can choose any state *a priori*.

The second local mapping is thus obtained by taking every local configurations (x, y, z) of $id_{\mathbf{s}}$ such that $id_{\mathbf{s}}(x, y, z) = A$, and setting them to D if $x = \star$, and to E otherwise. The result is not a deterministic space-time diagram, but this is easily corrected with two additional modifications for ACQ and QQC, leading to the local mapping depicted in the center of Figure 2. The space-time diagram on the right is obtained by applying the local mapping on the space-time on the left as indicated by the outlined local configuration on the left, and resulting state on the right, at the following timestep in direct application of Definition 5.

Proposition 11 *There is a n^3-RTSG solution using 5 states and 72 transitions.*

Proof

First note that the right space-time diagram is deterministic. We can therefore extract the transition table given in Figure 3 from it. No additional transitions appear after the space-time shown in Figure 2. To prove this CA to be an n^3-RTSG solution, it is enough to check that the local mapping is RTSG-compliant, and since the source CA is an n^3-RTSG solution, we can conclude using Proposition 10.

To ease the comparison of this 5-state solution with the original 6-state solution, the transitions that are different, added or removed are highlighted in the above table. Of course, all transition containing A should be considered as removed. The reader can check that these differences do not correspond exactly to those described in the local mapping.

3.3 Optimizing Through Millions of Solutions

Now that we have a first 5-state solution, we are ready to generate millions of them. The generated solutions are essentially the same, but can have fewer states or/and a different number of transitions. We begin by a brief summary of the algorithm (more details in [8]) and then examine the results.

3.3.1 The Exploration Algorithm

The exploration algorithm is related to the hand-crafted process above. In our case, we make it begin with an identity local mapping of the 5-state hand-crafted solution and let it explore many of its possible modifications by applying one modification at a time. Only compliant modifications are considered. In fact, this is a graph exploration algorithm, the node of this graph being compliant local mappings from the hand-crafted 5-state solution to a fixed set of 5 states (and an outside $\star$ state). The neighbors of a local mapping ℓ are all the local mapping obtained by exactly one compliant modification on ℓ. The identity local mapping is obviously a compliant local simulation, and the algorithm generates all its neighbors and add to the "remaining tasks" queue any neighbor that is also a compliant simulation. Continuing in this way with the content of the queue, the algorithm explores the complete connected component of compliant simulations. By Proposition 10, all these compliant local simulations are n^3-RTSG solutions. Let us describe two additional ingredients.

The first one is that there is an initialization step. To check that a local mapping ℓ is a local simulation, we need generate its local transition relation δ_{Φ_ℓ} to check if it is a function or not. This is easy to do for all the local mappings if we first collect all the *super-local transition* of the hand-crafted solution, *i.e.* all quintuplets of states with their resulting triplet of states appearing anywhere in the space-time diagram of the hand-crafted solution. From these data, and for any local mapping ℓ, it is enough to apply $\ell_{\mathtt{s}}$ on all the super local transitions to generate the entries of associated local transition relation δ_{Φ_ℓ}.

The second ingredient is that a parameter k to allow the discovery of more compliant simulation connected component. Indeed, with $k = 0$, the algorithm is unchanged and a compliant local simulation is reached only if its modifications can be applied one at a time while leading to compliant local simulation all the way through. With $k \geq 1$, the algorithm randomly apply k additional modifications simultaneously on any given compliant local simulation. If fact, it is often the case that many modifications need to be applied simultaneously, for example the two last modifications described in the design of the hand-crafted solution. So there is clearly room for improvement in the algorithm in this regard.

3.3.2 Generated Solutions and Optimizations

Running the algorithm on a 32 cores of 2.00GHz machine having 126Gb of memory, we obtain so many solutions that the algorithm stops because it runs out of memory resource. The first time, we ran the algorithm with $k = 0$. The program actually uses 2 cores and about 43 Gb of memory. We did not optimize the program nor did we check the configuration of the Java Virtual Machine for this Java implementation. Since the machine is shared, the following data are not really reproducible, but gives an idea of the execution.

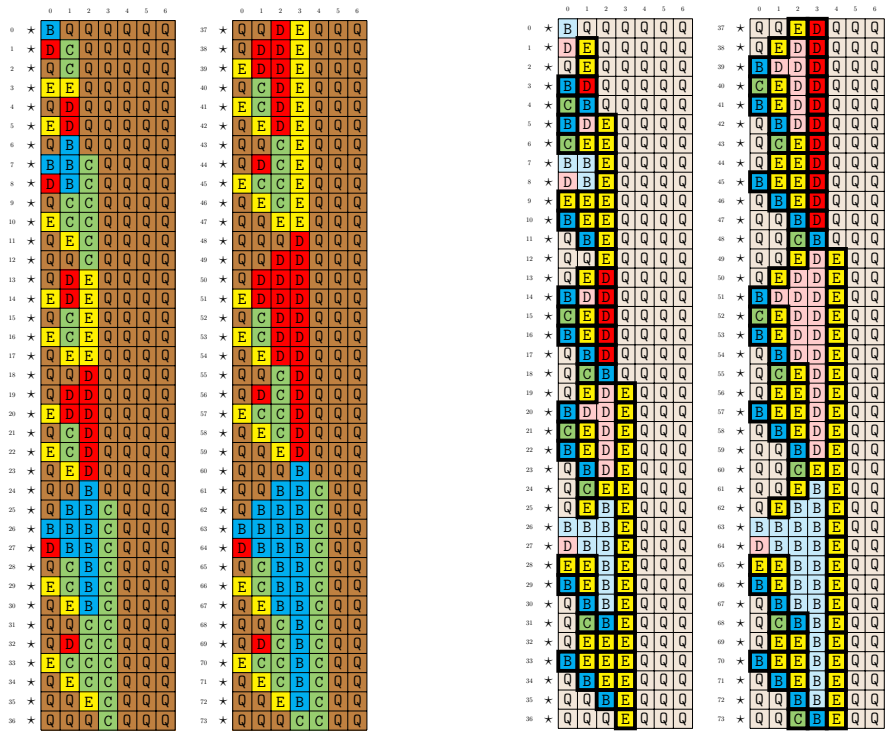

Figure 4: Hand-crafted 5-state diagram and optimized 5-state diagram using 58 transitions.

- after 1 days, about 15 millions local simulations.
- after 6 days, about 85 millions local simulations.
- after 20 days, about 90 millions local simulations.

The number of solutions found each day was steady for the 6 firsts days then dropped, presumably because of memory issues. Running concurrently the program with $k = 2$, it uses 2 cores and 36 Gb of memory before it stops because of the same lake of memory.

- after 1 days, about 15 millions local simulations.
- after 6 days, about 70 millions local simulations.
- after 20 days, about 74 millions local simulations.

In fact, we had to keep in memory all the solutions and check whether we obtain new solutions up to permutations, in order to be able to have a total number of generated solutions. Better strategies can be found if the goal is only to optimize the solution.

Proposition 12 *There are at least 90,000,000 n^3-RTSG solutions using 5 states.*

Among these millions of solutions, no 4-state solutions are found, but 32379 of them have fewer transitions. In the following table, the first line indicates a

Q	Right State				
Left State	Q	B	C	D	E
Q	Q	Q	Q		E
B	E				
C					
D	Q				
E	Q				
⋆	Q	Q	Q		B

B	Right State				
Left State	Q	B	C	D	E
Q		C		C	Q
B		B			B
C	D	E			E
D		E			E
E		B			B
⋆	D	D		C	Q

C	Right State				
Left State	Q	B	C	D	E
Q		E			E
B					
C					
D					
E					
⋆		B			B

D	Right State				
Left State	Q	B	C	D	E
Q					
B	B			E	E
C					
D	D			D	D
E	D			D	D
⋆		E			Q

E	Right State				
Left State	Q	B	C	D	E
Q	D	B		D	E
B	E	B		B	B
C				E	B
D	E				
E	E	E		E	E
⋆					B

Figure 5: Transition table of the generated 5-state solution using 58 transitions.

number of transition and the second line the number of solutions having this number of transitions.

58	59	60	61	62	63	64	65	66	67	68	69	70	71
1	7	22	51	98	174	336	589	1044	1618	2696	4643	7671	13429

Proposition 13 *There is a n^3-RTSG solution using 5 states and 58 transitions.*

Proof

The transition table of the generated solution is shown in Figure 5. The local mapping having 45 entries different from the identity local mapping, it is not very practical to display it as the previous one, but it is possible to reconstruct it from both cellular automaton. It is then a matter of checking that it is compliant and apply Proposition 10 to conclude as before.

3.4 Beyond RTSG and FSSP Optimizations

It should be clear by now that the approach can be applied to a large class of problems. For example, the same algorithm used here for the n^3-RTSG problem is not particular to the n^3 sequence and can be used for any sequence S, as clearly indicated in the definitions and propositions above. Also, the slightly differently parameterized algorithm for the minimal-time FSSP is not particular to minimal-time solutions and can be used for any synchronization time, without even specifying this synchronization time to the algorithm. The difference in the parameter only reflects the slightly different notion of compliance for RTSG problems and FSSP. Because the notion of compliance is the only changing factor, the approach can readily be adapted to any class of problem for which an appropriate notion of compliance can be designed. As examplified here, and in the FSSP case described in [8], the compliance property is a direct translate of the problem.

4 Conclusion

There are still many components of this work to communicate properly, including how local mappings compose and relate to each other and how the integration of non-deterministic family of space-time diagrams can allow to explore even more (deterministic) solutions. Beginning these discussions in this conclusion is not necessarily useful. There is nonetheless one aspect on which we should comment. The notion of local mapping appears to be a bridge between a common practice and a topological tool. Indeed, on the practical side, it is common to work directly at the level of space-time diagrams, and this is practice that is captured formally, and only partly, by local mappings. This allows to automate this practice. On the other hand, a question was raised about the relation with conjugacy classes, a standard notion in the cellular automata and symbolic dynamics literature [2]. In fact, the concept of local mapping appears to be an adaptation of the notion of shift-equivariant homomorphism between two cellular automaton. Such homomorphisms are usually described on total transition functions, with any configuration being a valid initial configuration. This is a dynamical system point of view not necessarily aligned with the more algorithmic point of view of FSSP and RTSG problems. Local mappings augment the notion of homomorphism by including the partiality of the transition functions and the temporal aspect of the space-time diagrams, essential for the very specification of many algorithmic problem. Forming a bridge between the algorithmic and dynamical points of view might be the reason of their effectiveness.

References

[1] Patrick C. Fischer. Generation of primes by a one-dimensional real-time iterative array. *J. ACM*, 12(3):388–394, 1965. doi:10.1145/321281.321290.

[2] Joonatan Jalonen and Jarkko Kari. On the conjugacy problem of cellular automata. *Inf. Comput.*, 274:104531, 2020. doi:10.1016/j.ic.2020.104531.

[3] Naoki Kamikawa and Hiroshi Umeo. A Study on Sequence Generation Powers of Small Cellular Automata. *SICE Journal of Control, Measurement, and System Integration*, 5(4):191–199, January 2012. doi:10.9746/jcmsi.5.191.

[4] Naoki Kamikawa and Hiroshi Umeo. A construction of five-state real-time fibonacci sequence generator. *Artif. Life Robotics*, 21(4):531–539, 2016. doi:10.1007/s10015-016-0309-2.

[5] Naoki Kamikawa and Hiroshi Umeo. Two implementations of real-time sequence generator for {n^3 | n=1, 2, 3, ... } and their comparison. *Int. J. Netw. Comput.*, 9(2):257–275, 2019. URL: http://www.ijnc.org/index.php/ijnc/article/view/209.

[6] Ivan Korec. Real-time generation of primes by a one-dimensional cellular automaton with 9-states. In Maurice Margenstern, editor, *International Colloquium Universal Machines and Computations, MCU'98, Metz, France, March 23-27, 1998, Proceedingsi, Volume II*, pages 101–116. IUT Metz, 1998.

[7] Naoki Kamikawa and Hiroshi Umeo. Some algorithms for real-time generation of non-regular sequences on one-bit inter-cell-communication cellular automata. In *SICE Annual Conference 2007*, pages 953–958, 2007. doi:10.1109/SICE.2007.4421122.

[8] Tien Thao Nguyen and Luidnel Maignan. Exploring millions of 6-state FSSP solutions: The formal notion of local CA simulation. In Hector Zenil, editor, *Cellular Automata and Discrete Complex Systems - 26th IFIP WG 1.5 International Workshop, AUTOMATA 2020, Stockholm, Sweden, August 10-12, 2020, Proceedings*, volume 12286 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2020. doi:10.1007/978-3-030-61588-8_1.

[9] Tien Thao Nguyen and Luidnel Maignan. Some cellular fields interrelations and optimizations in FSSP solutions. *J. Cellular Automata*, 15(1-2):131–146, 2020. URL: https://www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-15-number-1-2-2020/jca-15-1-2-p-131-146/.