Second-Order Rules on Elementary Cellular Automata

by Atell Krasnopolski

2021

Abstract. Cellular automata are a famous example of discrete dynamical systems. However, more dynamics can be added to such systems by introducing second-order rules to modify the base rules of automata. The simplest example of this is using basic binary operations on Wolfram numbers of elementary cellular automata (ECA) on each step. This may sound very simple, yet it provides us with a wide variety of tools to manipulate ECA.

1. The First-Order and Second-Order Rules

In elementary cellular automata (ECA), it is usually convenient to use Wolfram numbers to describe different rules with a single number. It is assumed that you know how are these numbers constructed and what meaning do they have. Throughout this paper, I will only cover these basic `0-255` ECA rules for simplicity.

The point of SOECA (Second-Order ECA) is to apply a certain deterministic rule modification on each step of the system. Starting with a base rule, we will get a different one every step. Here are simple modifications (with their short notation) that we can apply:

```
> --- circular bit shift to the right
< --- circular bit shift to the left
+ --- increment the rule
- --- decrement the rule
! --- perform NOT on each bit
| --- flip rule bit-wise
```

Note: `+` and `−` are circular arithmetic operators on 8-bit integers, which means that adding 1 to 255 would result into 0.

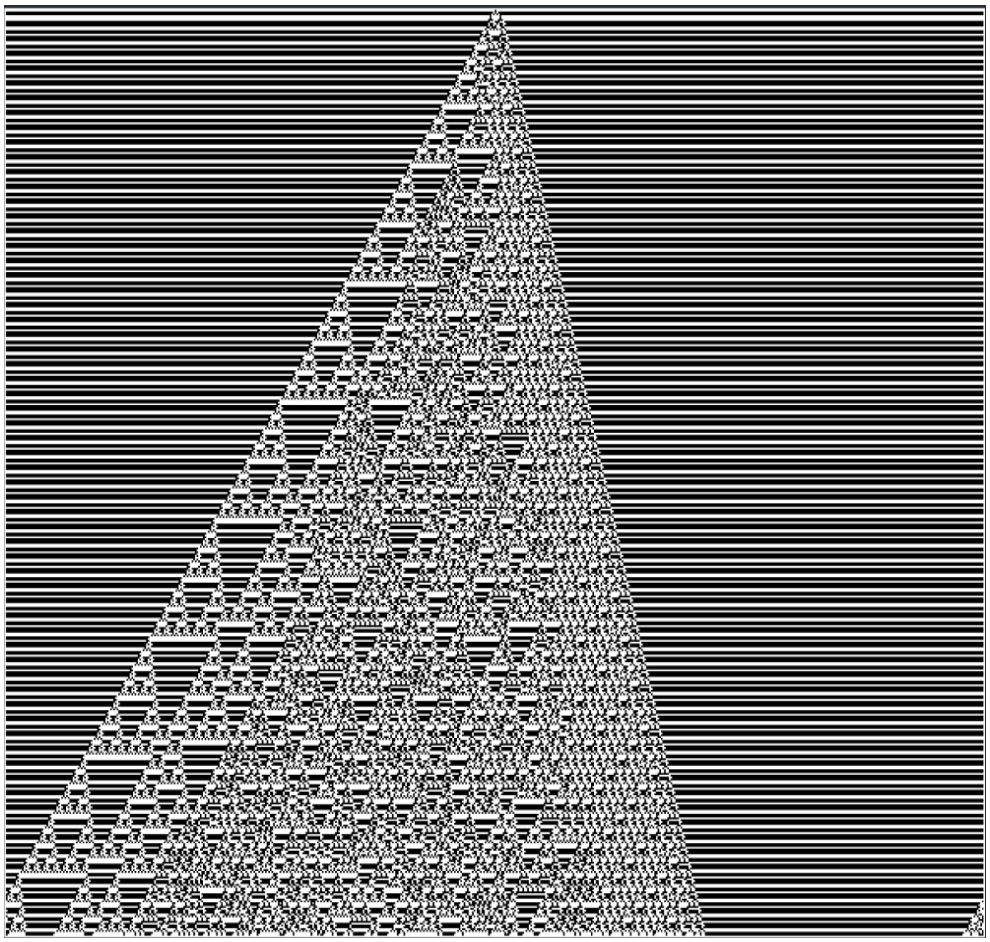

Figure 1: Rule 101> with a singular active central cell as the initial state

It is possible to use these modifications on their own (see fig.1) as well as to combine them in a sequence of operations on the rule. This sequence is essentially **the second-order rule** itself. Obviously, second-order rules are not associative (meaning that, for instance, "<+" and "+<" are not the same). Note that, for example, ">+<" is equal to "++" as "<>" is equal to "" (empty rule).

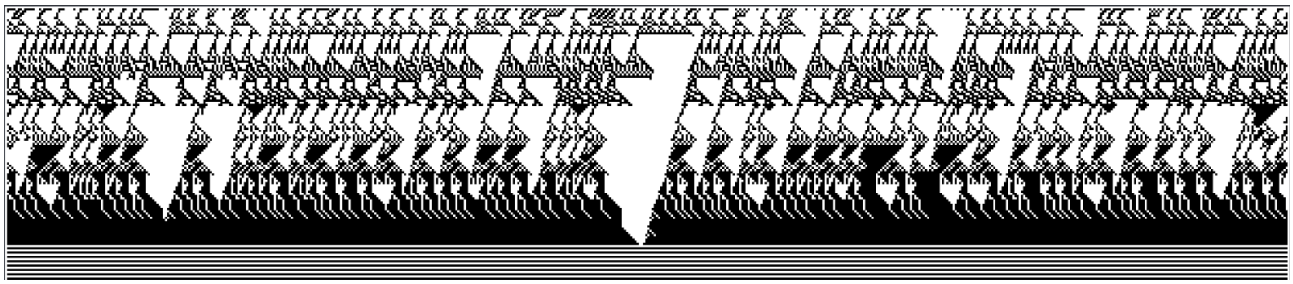

Figure 2: Rule 40++ with a random initial state (more on this pattern later)

2. Thermodynamics, Entropy, and Chaos

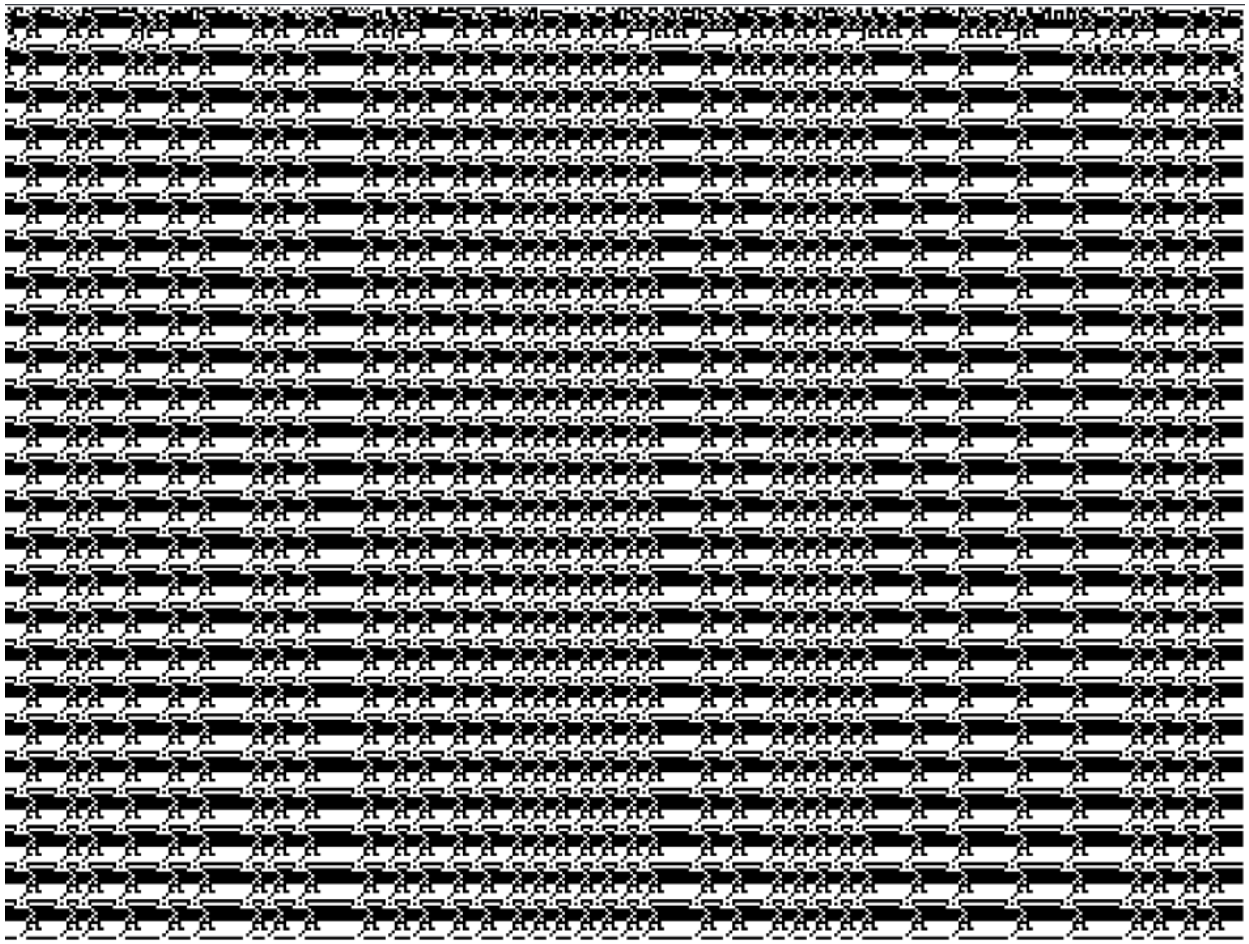

Figure 3: Rule 105>>+ with a random initial state

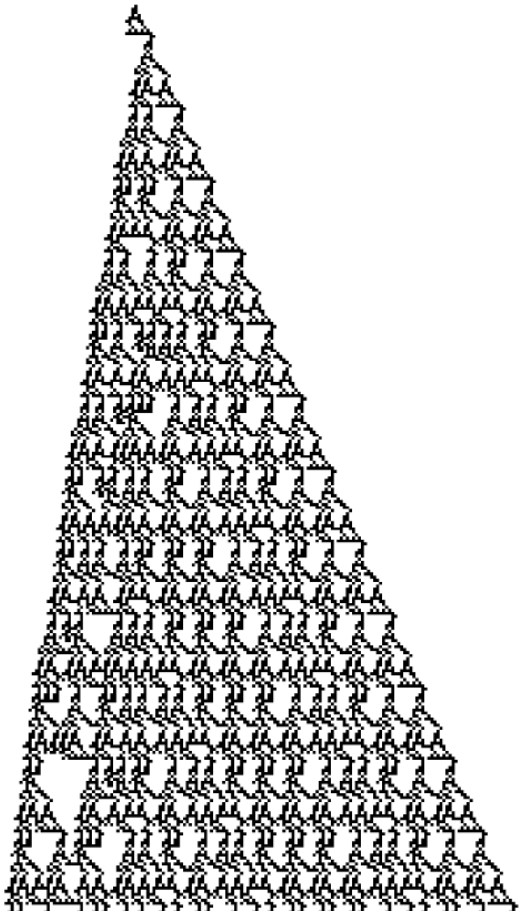

Figure 4: Rule 110-|+!

As most of the ECA, SOECA do not behave according to the second law of thermodynamics by reducing entropy from a random distribution of cells to the emergence of certain patterns. One of the best illustrations of this that I have discovered is rule `105>>+` (see fig.3) which will always end up with the same multi-line pattern repeating regardless of the initial distribution. Moreover, the initial distribution here only controls the number of "segments" in the pattern. Obviously, this automaton is not reversible. Thus, this behaviour does not contradict the law, as the law can only be applied to reversible systems.

Many other rules, such as `105>>>` or `110-|+!` (fig.4), on the other hand, do produce some chaotic shapes starting with a central active cell.

3. Repetitive (Periodic) Automata

If an automaton repeats its states periodically (fig.3), we call it a **first-order repetitive (or periodic) automaton**. Such automata would have Wolfram class 2. If an automaton repeats its rules periodically, we call it a **second-order repetitive (or periodic) automaton** or **rule-repetitive automaton**. This characteristic is only dependent on the second-order rule itself. Important to notice that an automaton can be second-order repetitive and not first-order repetitive. An example of this would be rule `110-|+!` (fig.4) that is not state-repetitive but clearly follows a cyclic rule pattern:

```
110 → 72 → 28 → 38  → 90  → 100 → 56  → 18  → 118 →
80  → 12 → 46 → 74  → 108 → 40  → 26  → 102  → 88 →
20  → 54 → 82 → 116 → 48  → 10  → 110 → ...
```

The rule `0>>+` (and consequently any `A>>+` rule) is not periodic because it will never go back to rule `0` (or `A`, in the general case).

4. The "++" Pattern and Tree-like Shapes

As shown by figure 2, the `40++` rule has a very intriguing layered pattern to it. This can be generalized to any rule `A++` where `A` is between `0` and `255`. If we start with one active cell in the initial state, it becomes clear that actually, the pattern here is very much like a tree, with some branches and layers (fig.5). When we start with a random distribution, these branches can overlap,

which results in more complex behaviour. Some rules, however, result in result in longer single branches than the others. For example, rules `0++` and `8++` result into branches of length 0 (where length is simply the number of iterations the pattern holds), although rule `8++` can give us longer branches if we start with a random distribution of active cells. Such patterns cannot be repetitive or last forever, since at some point we will get either rule `0` or rule `255`.
Note: other second-order rules besides `"++"` can also generate similar patterns, see rule `40+`.

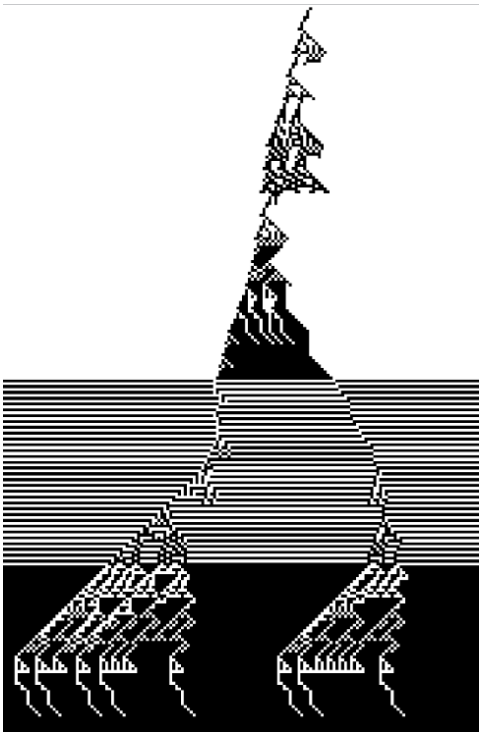

Figure 5: Rule 2++; the longest possible "++tree" of length 243

5. The Effects of `"A!"`

One of the most property-preserving second-order rules is `"!"`. For any rule `A`, `A!` has the same Wolfram class and often even follows a similar pattern. Rules `30` and `30!` can be a good example of this (fig.6). Though it may not be obvious at first glance, these rules are very similar in their structure: expanding ordered pattern on the left and expanding chaos on the right. These two areas are separated by a chaotic imaginary line in both.

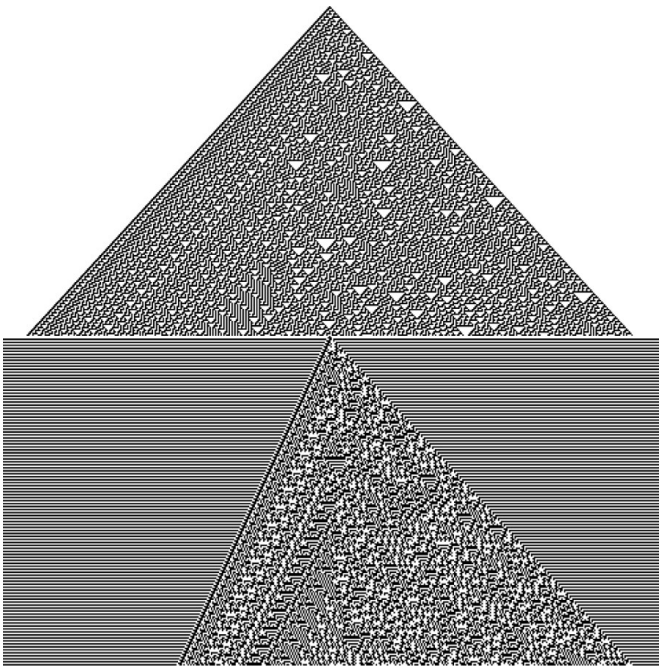

Figure 6: Rule 30 (top) and Rule 30! (bottom)

6. Fractal Shapes; “>>|” and “<<|” for Combining Rules

ECA are also known for producing fractal patterns. SOECA are no exception and frequently produce patterns similar to Sierpinski triangles (fig.7). This exact “distorted Sierpinski triangle” pattern occurs with the same second-order rules “>>|” and “<<|” and base rules 15, 60, 195, and 240 because these second-order rules are repetitive with period T = 1. Shortly speaking: base rules 60 and 195 are responsible for the triangular patterns, second-order rules are responsible for a specific rule alternation with rules 15 and 240, while these, in turn, shift the pattern to the right and/or provide colour negation.

These “>>|” and “<<|” (and those similar to them) second-order rules are, in fact, intriguing for their period T = 1. This is a tool for making alternations between two rules. In the case described earlier, it combines properties of the two rules that alternate. We can also combine famous ECA rules 60 and 30 by alternating them. Simple computations show that they are connected by the second-order rule “<|” with period T = 1. Therefore, rules 60<| and 30<| are combinations of them (fig.8).

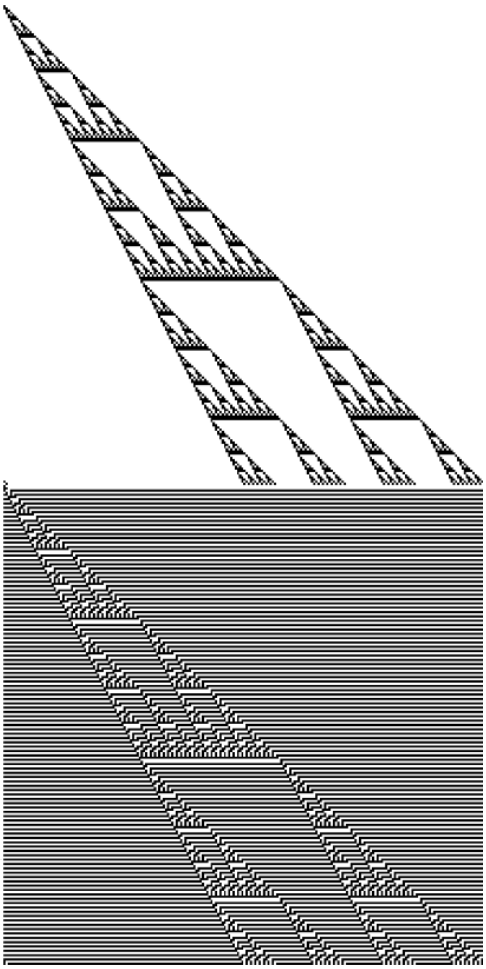

Figure 7: : Rule 60>>| or 60>|< (top); Rule 60<<| or 60<|> (bottom)

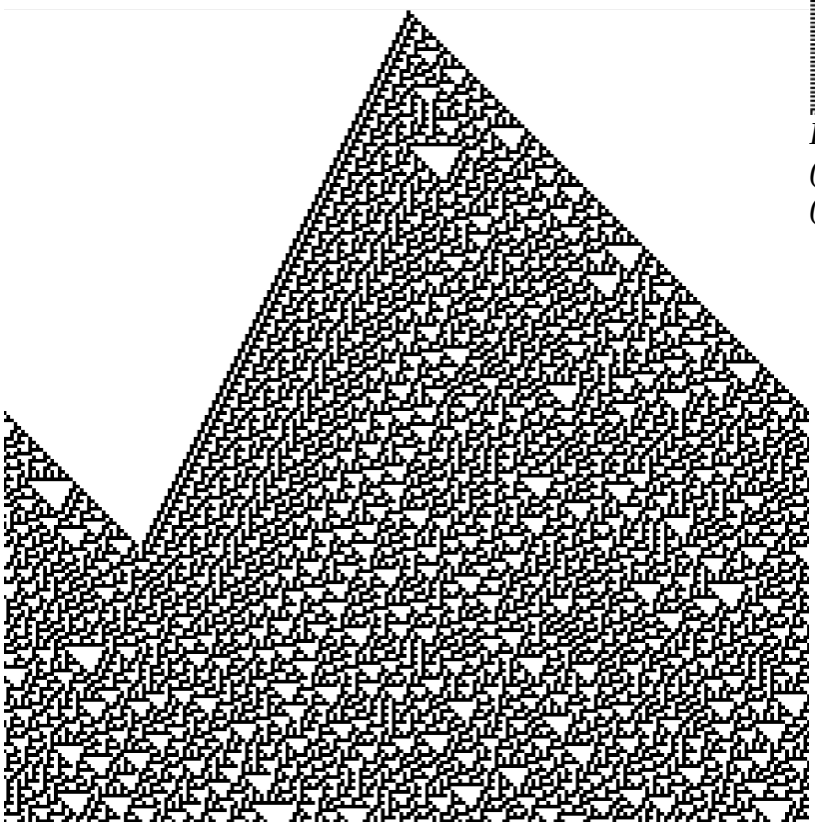

Figure 8: Rule 60<|

7. Turing Completeness

Turing completeness of specific SOECA with non-empty second-order rules is yet to be proven. However, since it is possible to construct a Turing-complete ECA rule, it should also be possible to make one using SOECA.