Elementary Cellular Automata with Memory

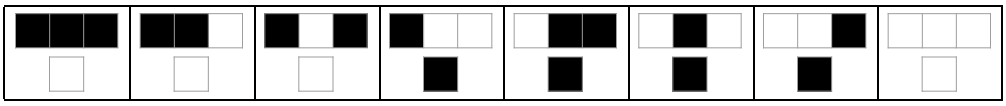
Paul-Jean Letourneau

Department of Physics, University of Calgary

Elementary Cellular Automata (ECA)

In order to explain how an Elementary Cellular Automaton with Memory works, let me put it in the context of an ECA.

In an ECA, one has a row of cells, each either black or white. Each cell is updated in parallel according to a simple rule that states what the new color of the cell should be, depending on its current color and that of its left and right neighbors.

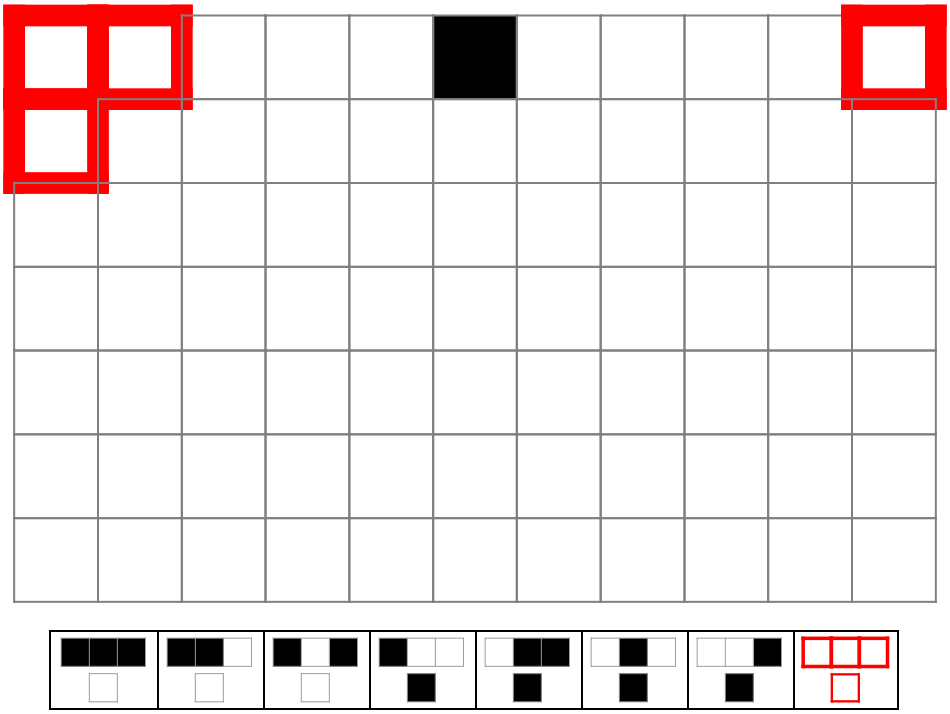


There are 256 such rules, but only 88 remain after removing cases that are equivalent with respect to left-right reflection and black-white inversion.

Elementary Cellular Automata cont'd

Although all the cells in a row are updated in parallel in a CA, this animation shows each cell being updated one at a time.

The red border shows the cell being updated, along with its neighborhood above it. In the rule table at the bottom, the rule case that's being used for that particular cell is also highlighted in red.

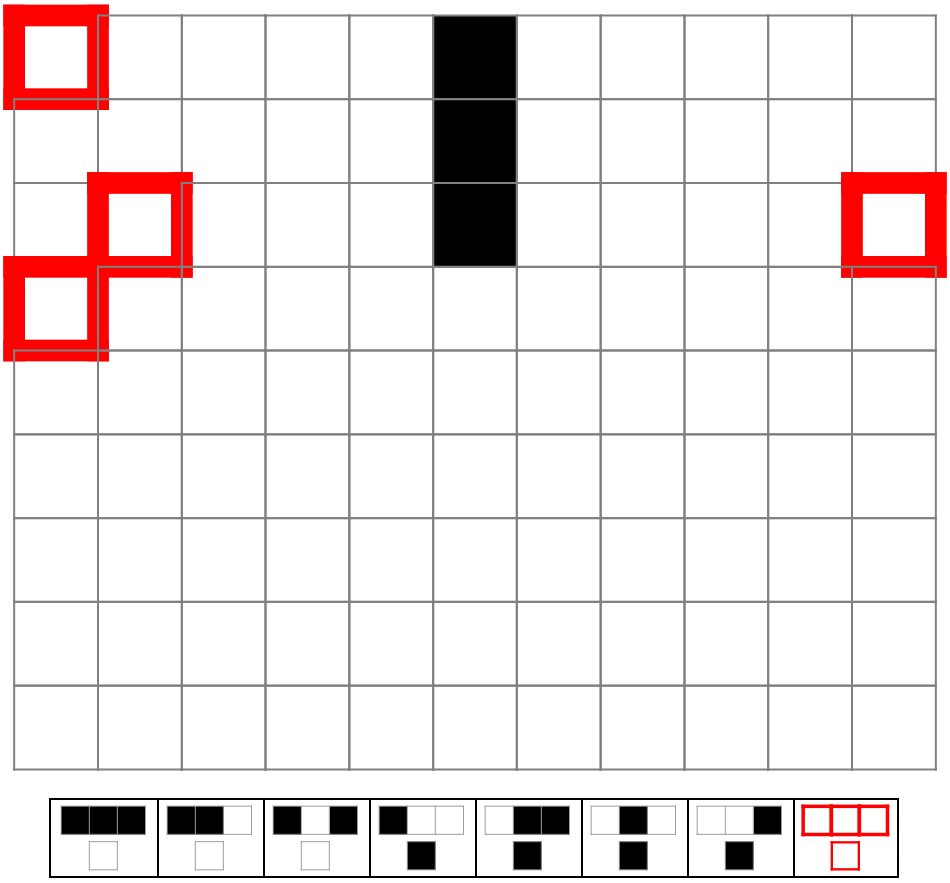


Elementary Cellular Automata with Memory

Here is how I propose to incorporate memory into an ECA.

Now the value of the centre cell in the neighborhood will be referenced from a step in the past.

Here is an example where the value of the centre cell is taken from two (2) time steps into the past. The "memory" here is therefore 2.



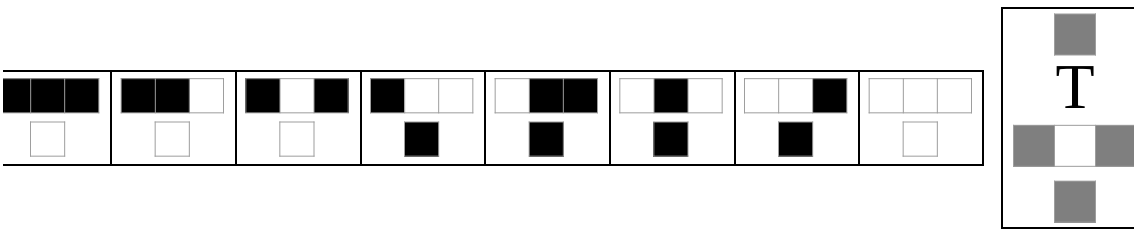
Notice that we're still using the same rule table. Everything is the same as in an ECA, except for the way we reference the centre neighborhood cell.

Also note that we have to specify more rows in the initial condition in order to accomodate the memory. In general, if the memory is T, then the initial condition has to have $T+1$ rows.

When $T = 0$, we recover the ECA again.

Elementary Cellular Automata with Memory: ECAM

Here is the full rule table for an ECA with Memory:



In addition to the ECA rule table we specify the memory in the centre cell.

Instead of being long-winded and always saying "Elementary Cellular Automaton with Memory T in centre cell", I'll just use the acronym ECAM-T (I pronounce it "eekam").

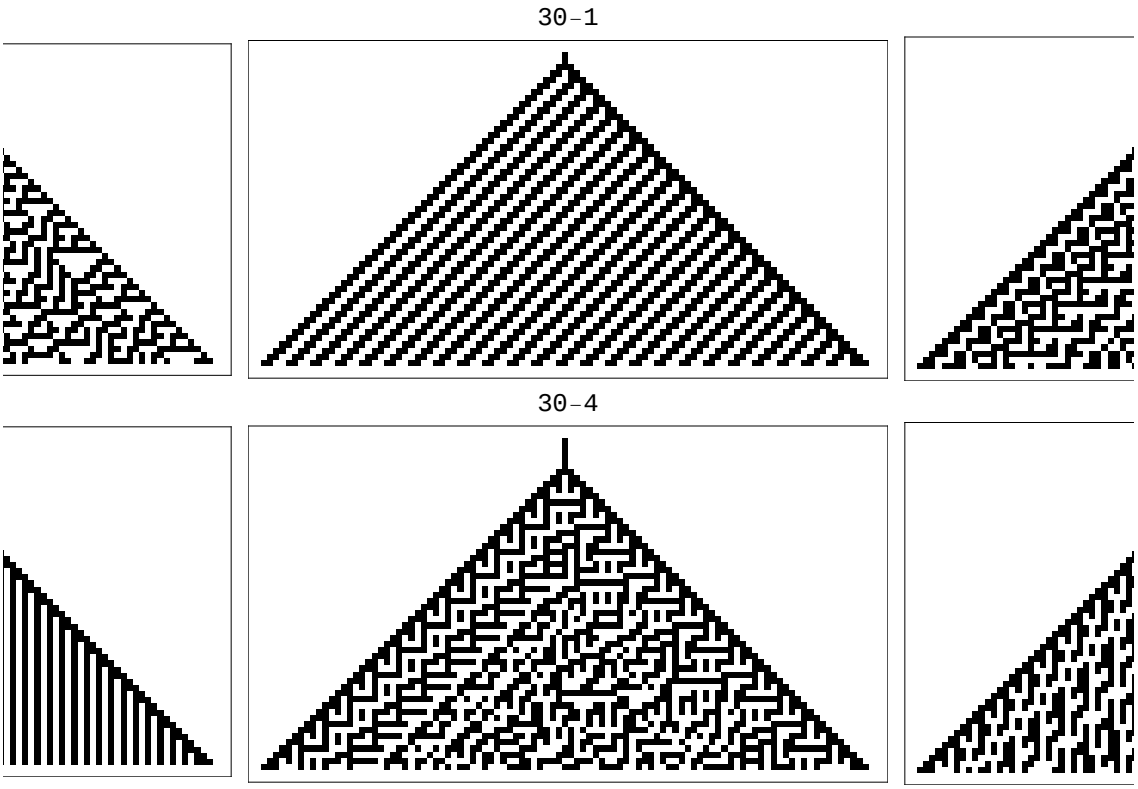
To be really compact when I am labelling figures and such, I'll also use the notation x-T, to mean "Rule x ECAM-T".

There are of course lots of ways one could imagine adding memory to a CA. But in keeping with the NKS philosophy, I have tried to keep it as simple as possible. In an ECAM, we are only considering ECA rule tables, and we are only time-shifting the centre cell.

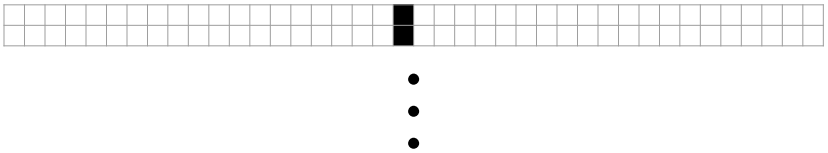
In fact, this memory scheme was suggested to me by Stephen Wolfram at the 2004 NKS Summer School, which is where this project got started.

Rule 30 ECAM-T

Here is rule 30 with memories from 0 (the ECA case) to 5:

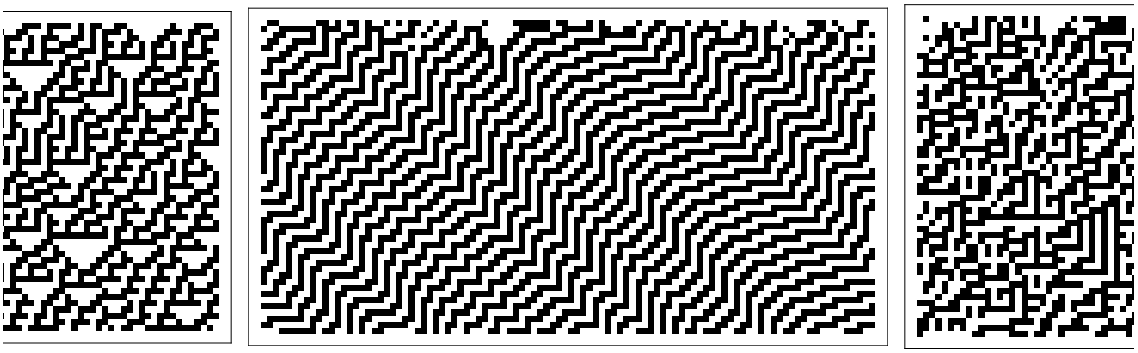


The initial condition being used here is $T+1$ rows with a single black cell in the middle:

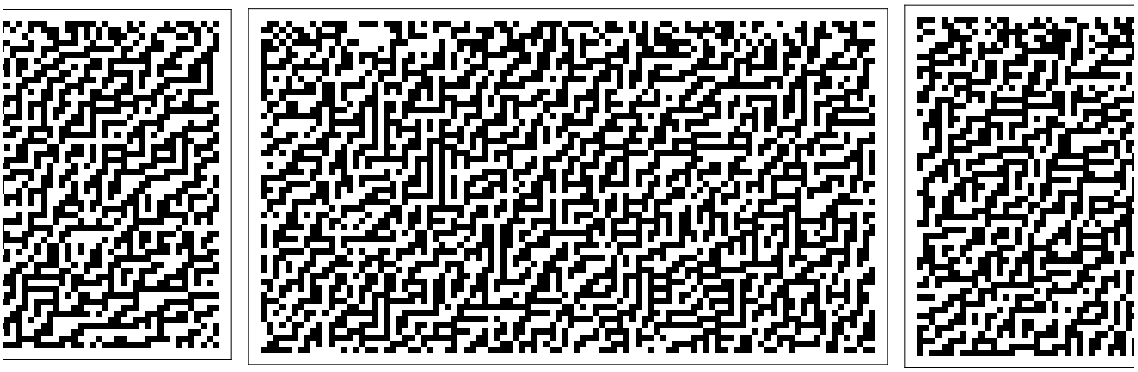


Rule 30 ECAM-T: Random Initial Conditions

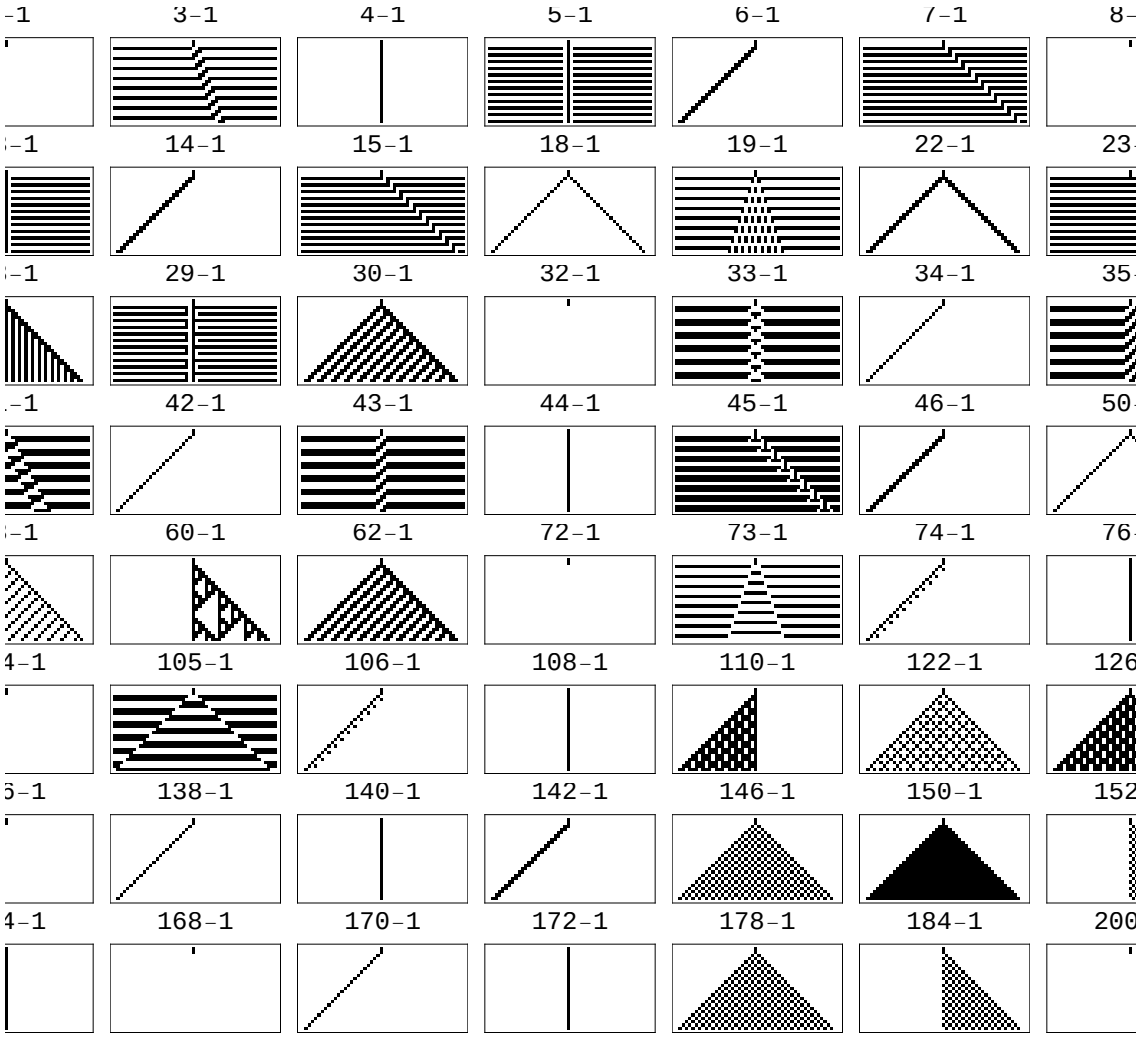
30-1



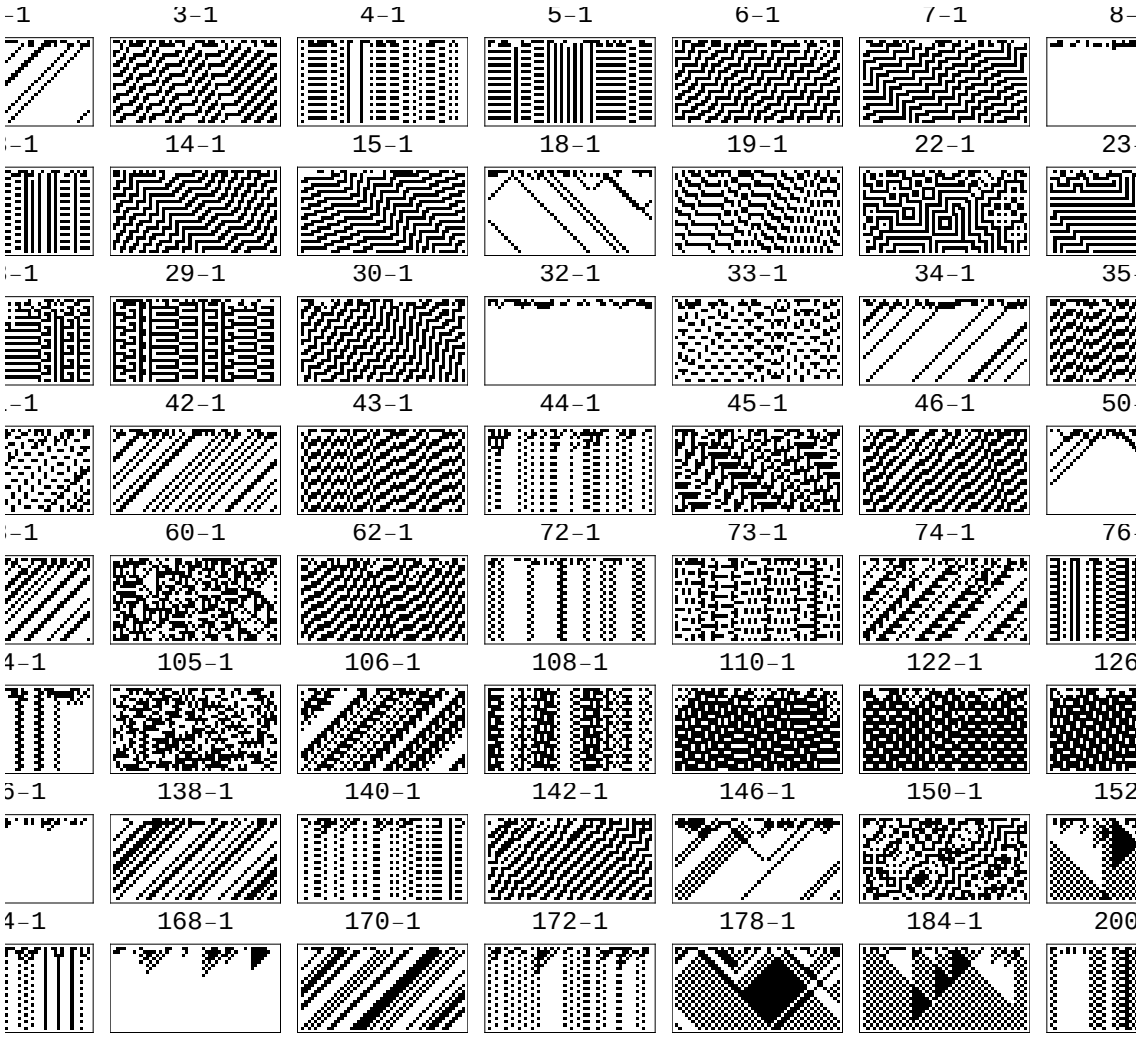
30-4



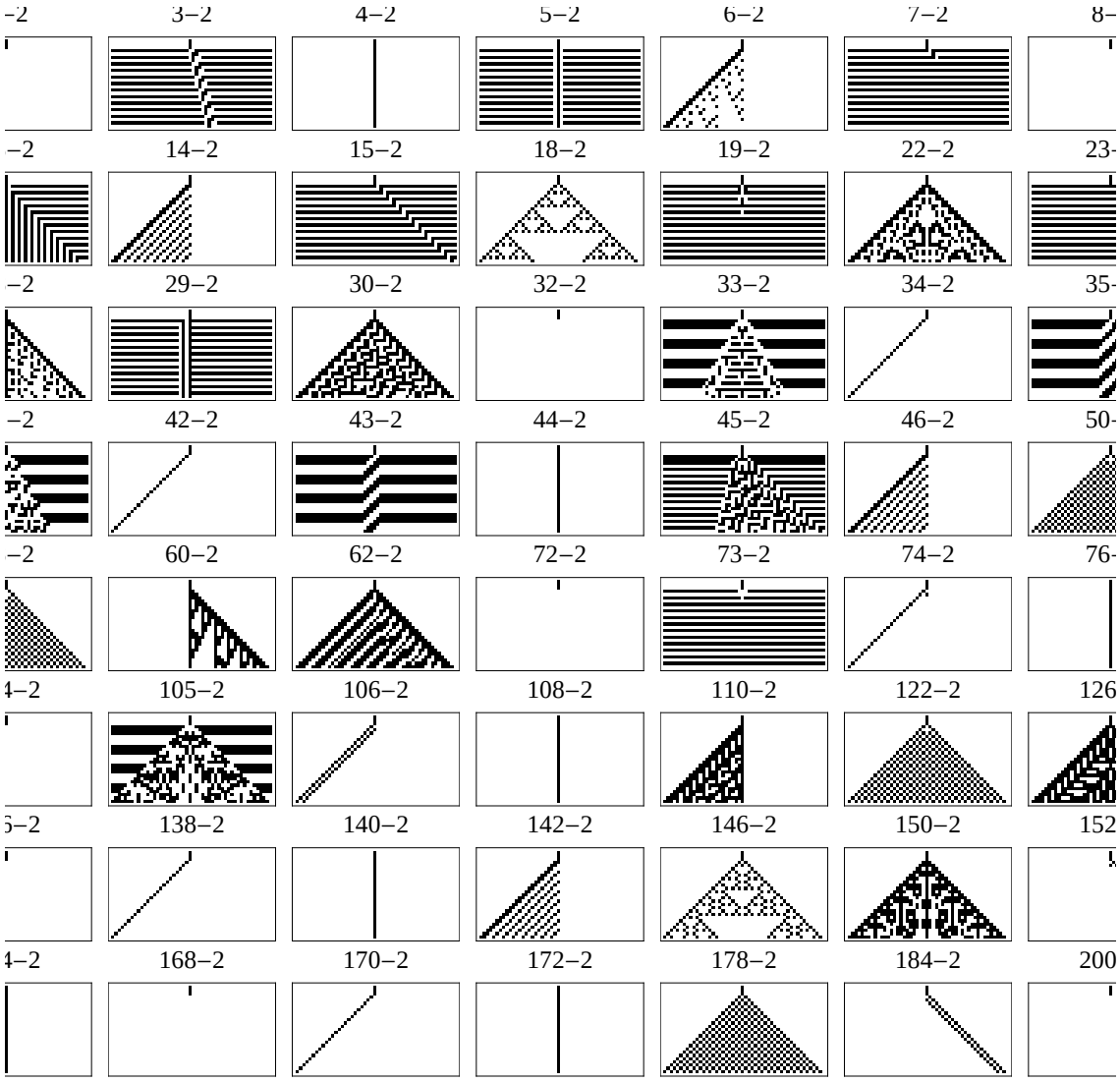
All ECAM-1: Simple Initial Condition



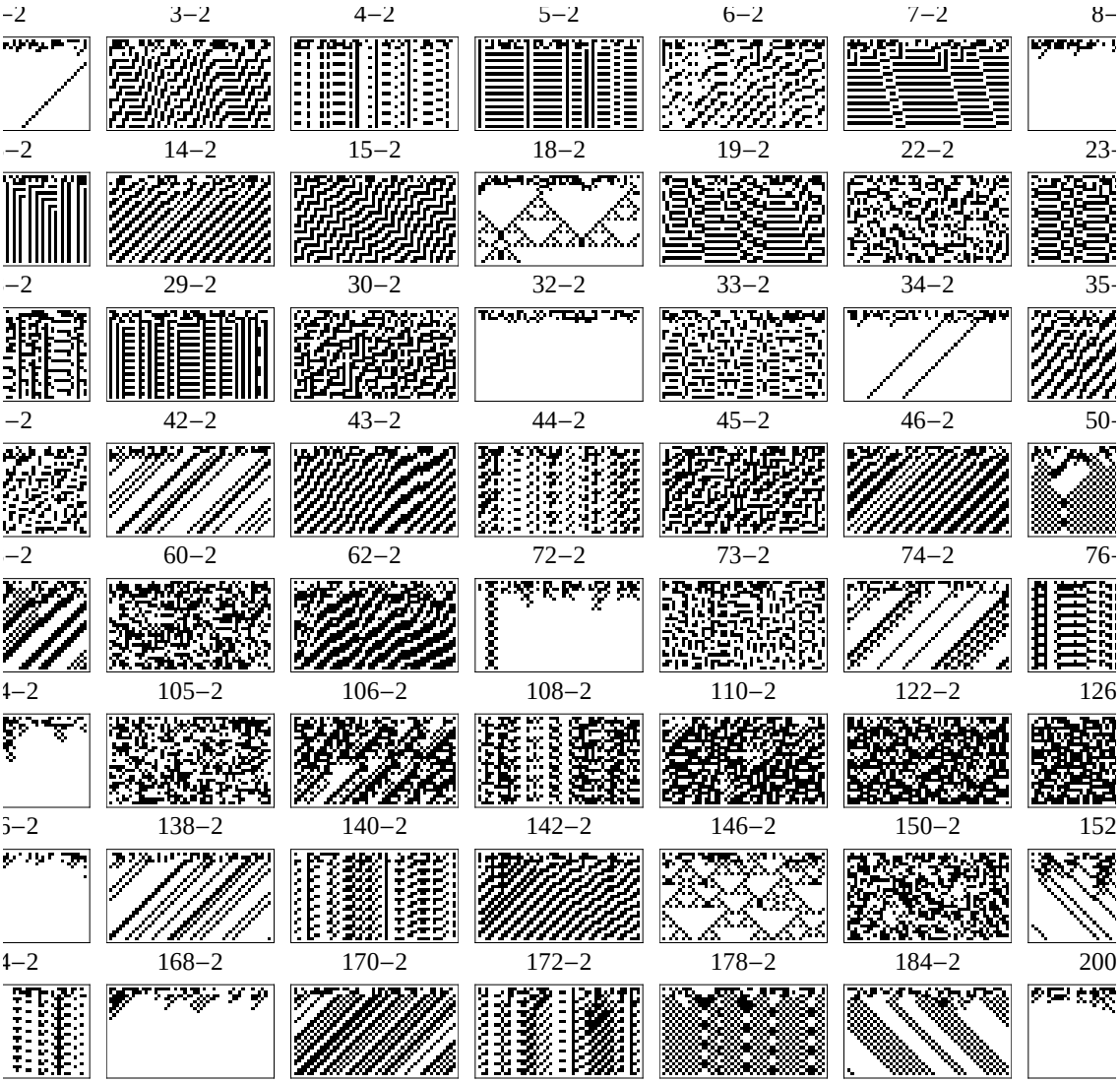
All ECAM-1: Random Initial Conditions



All ECAM-2: Simple Initial Condition



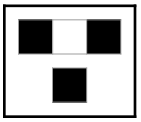
All ECAM-2: Simple Initial Condition



Emulating an ECAM with a Cellular Automaton

Before I show more of this corner of the computational universe, I'd like to show how to construct a Cellular Automaton that emulates an ECAM-*T*, for any memory *T*.

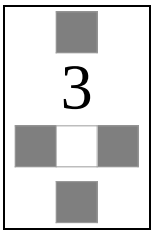
Let's start by considering a particular ECA rule icon:



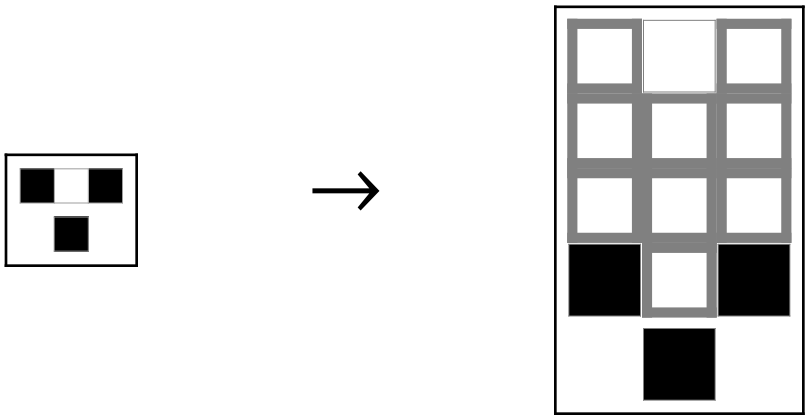
ECAM-3 Emulation

For concretness, I'll use an ECAM-3 as an example. It should be clear how it works for any memory T.

In an ECAM-3, the centre cell is referenced from 3 steps back in time:



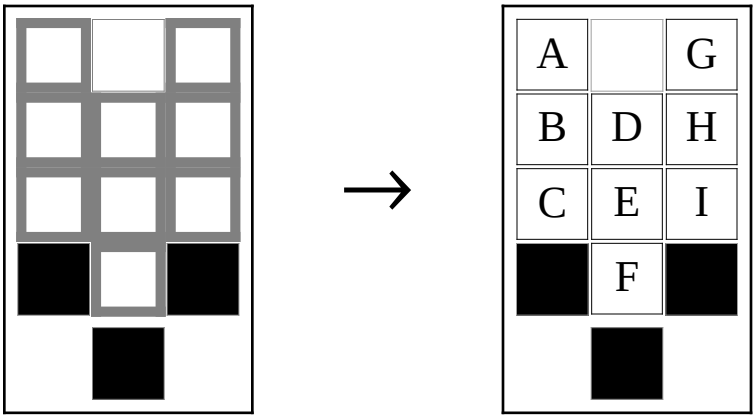
So the rule icon with intermediate time steps shown becomes



where the cells not being used in updating the cell are outlined in gray. Call them "don't-care bits", because the rule doesn't "care" about them.

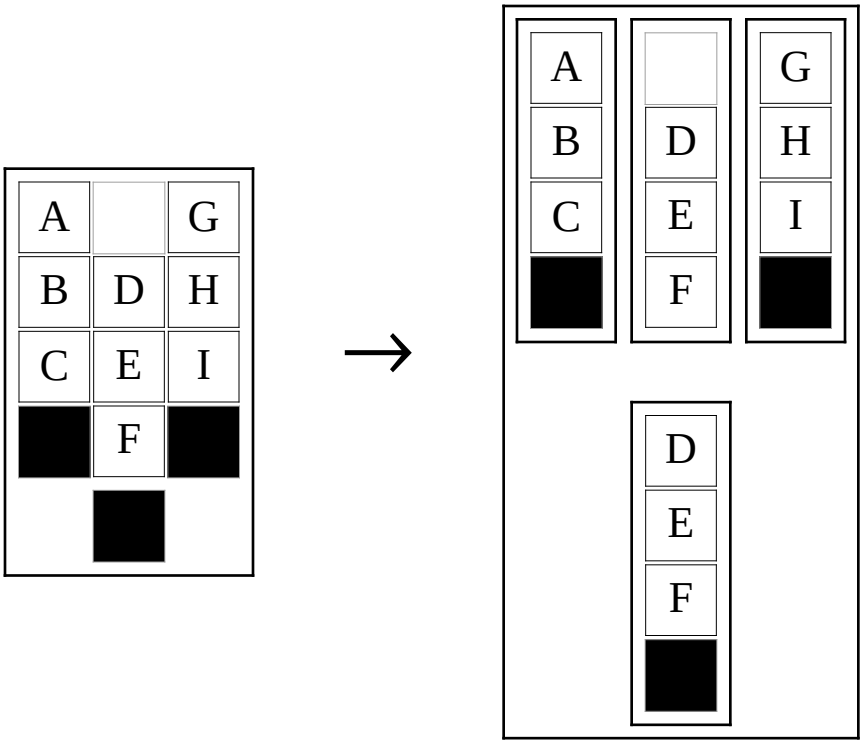
ECAM-3 Emulation

Now label the don't-care bits with letters:



ECAM-3 Emulation

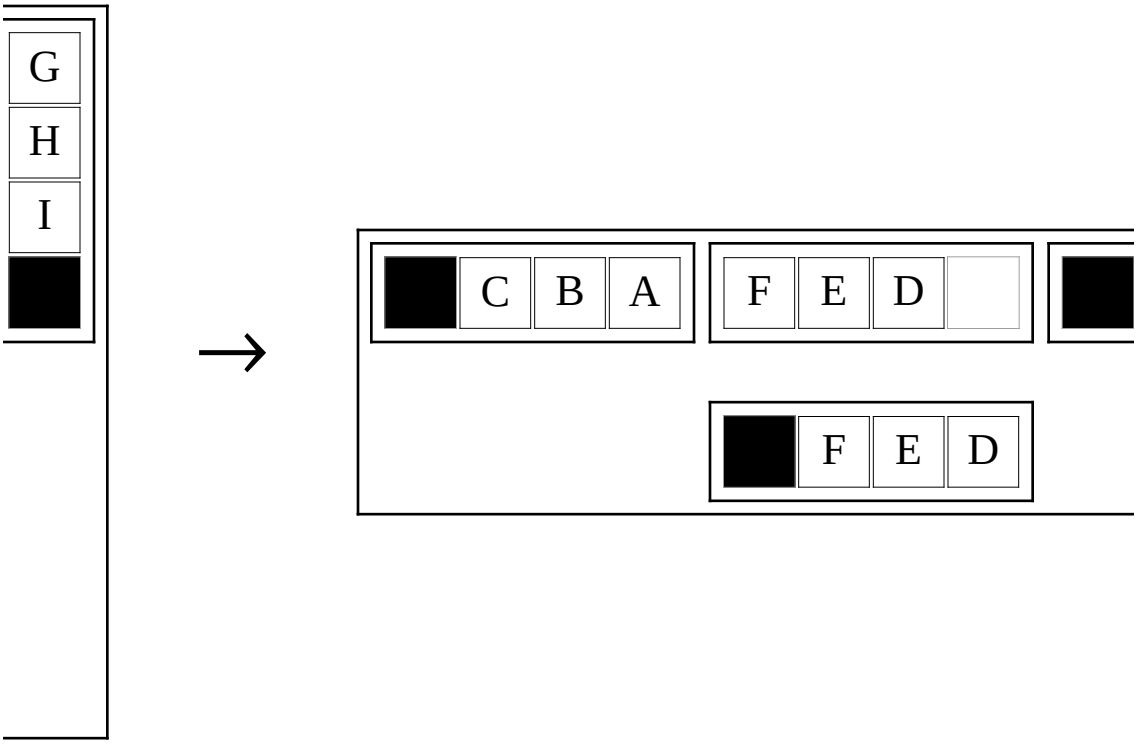
For each cell in the neighborhood, we keep track of 4 cells (in general $T+1$) in the following way:



Notice that for the output bit, we keep track of cells D, E, and F, which are the same don't care bits that we keep track of for the middle input bit.

ECAM-3 Emulation

We now encode the time histories for each cell as a base-2 digit sequence, where the most significant bit is the most recent in time.



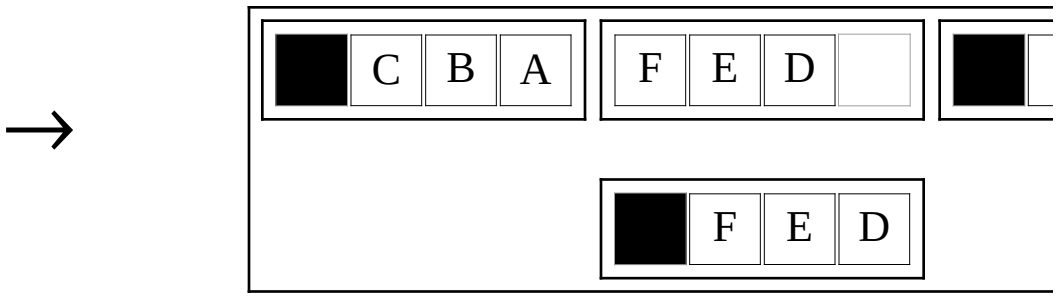
Therefore the time history for each cell is encoded as a single cell.

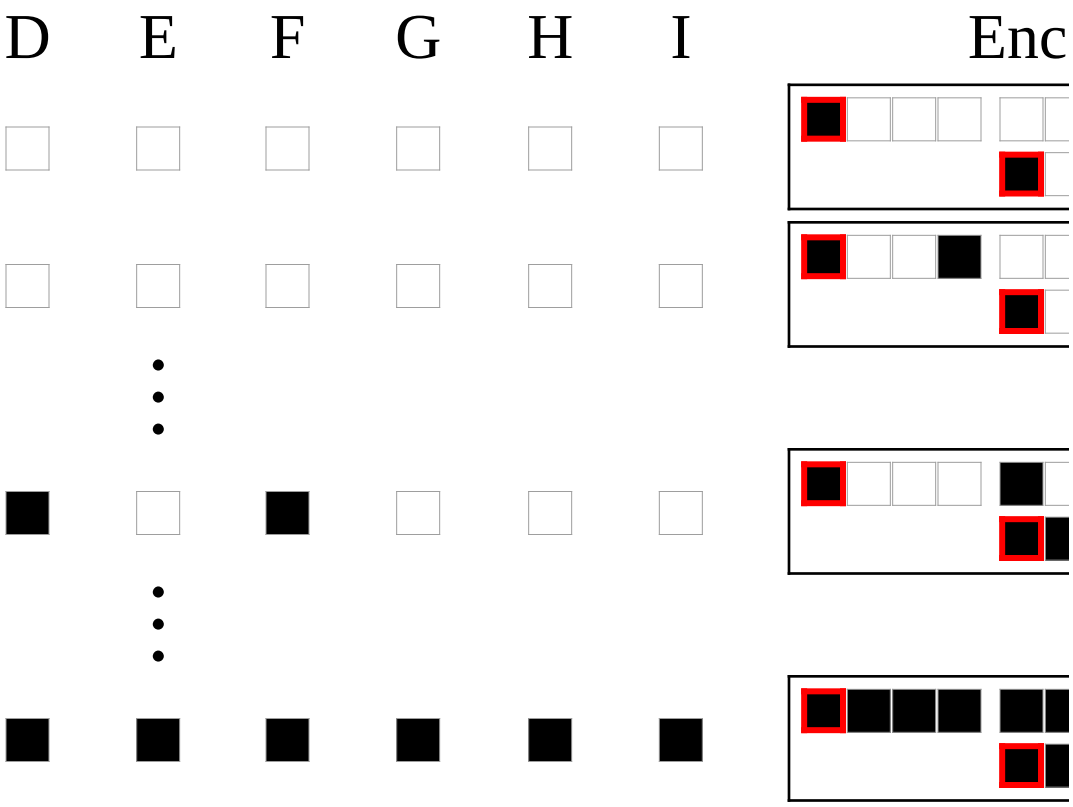
ECAM-3 Emulation

Now the crucial thing to recognize is that since the ECAM is totally independent of the values of the don't care bits A through I, the emulation must work for all possible values of these bits, for a given rule icon.

Therefore, for each ECA rule icon, we populate the rule table of the emulation with all possible combinations of don't care bits. The cells outlined in red below are the cells that stay the same, because they are fixed by the rule icon.

Note in the figure that as we take all combination of don't care bits, we always make sure that bits D, E, and F are preserved from the input to the output. This is what really enforces a consistent time-encoding.





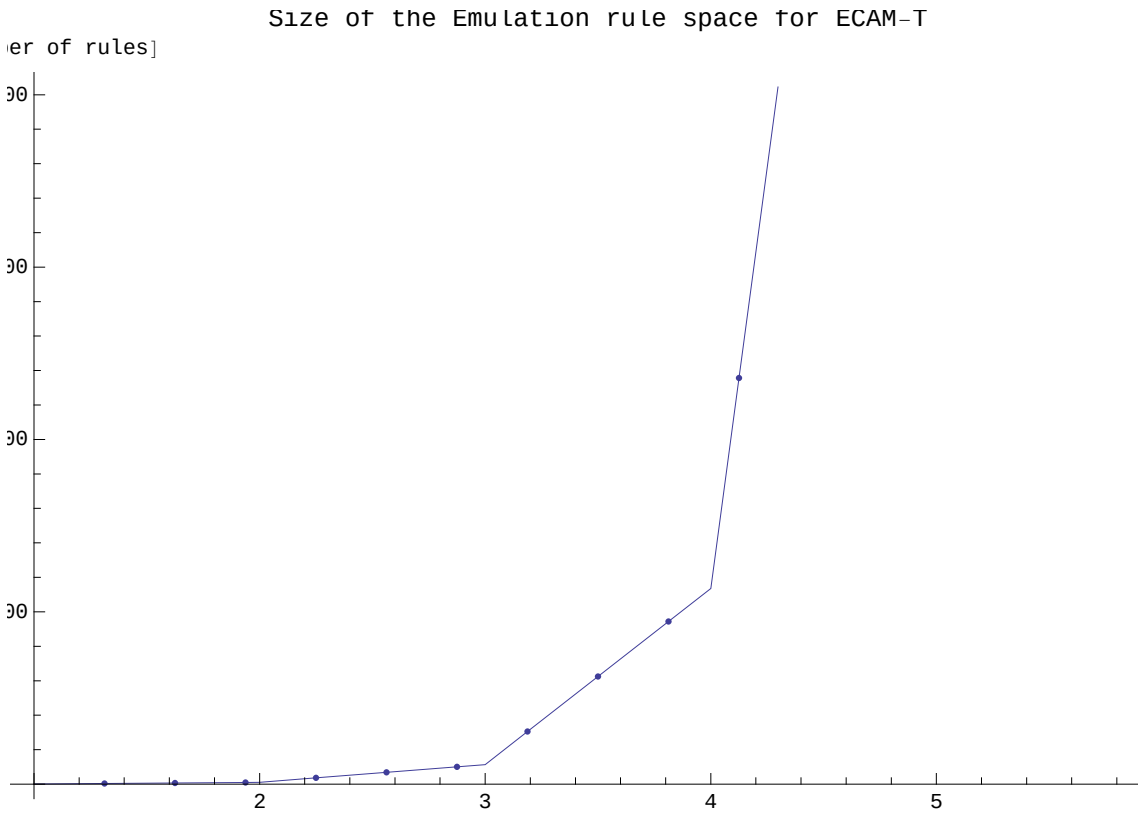
ECAM-3 Emulation

Since each cell in the emulation contains 4 bits, which can have $2^4 = 16$ different values, the emulation of an ECAM-3 requires a $k = 16$ color CA.

In general, this emulation generates a $k = 2^{T+1}$ color CA. So the emulations live in a rule space with $k^{k^3} = \left(2^{T+1}\right)^{\left(2^{T+1}\right)^3}$ rules. For ECAM-1, that's a rule space with

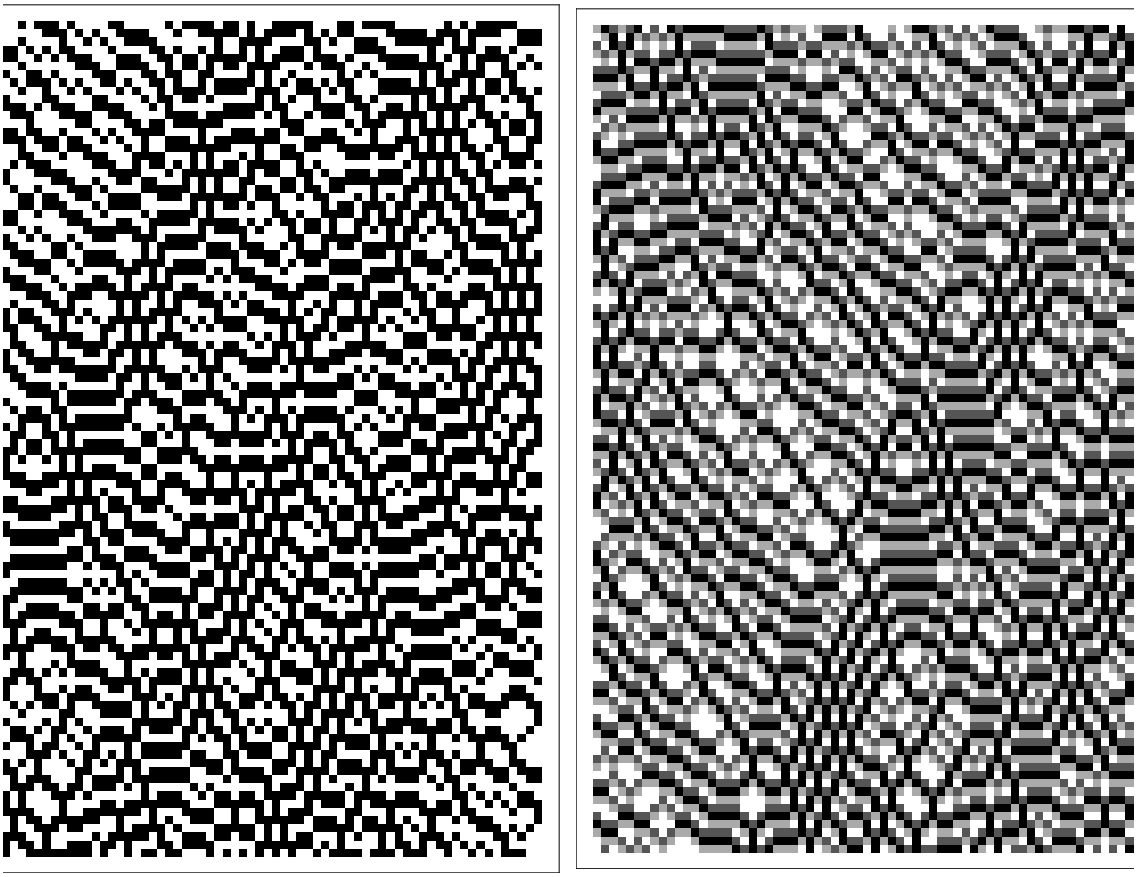
340 282 366 920 938 463 463 374 607 431 768 211 456

rules.



ECAM-1 Emulation: Example

Here is an example of a 4-color CA emulating rule 54 ECAM-1, from random initial conditions. The rule number of the 4-color rule is 114 308 428 836 046 362 728 713 265 888 850 414 240.

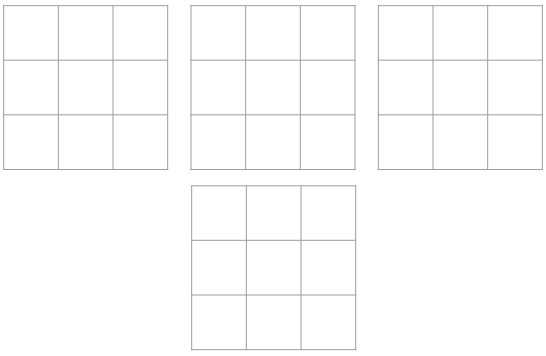


ECAM block emulation

It turns out that there is another kind of emulation that one can construct for an ECAM-T.

If one considers square blocks of cells of side length $T+1$, one finds that these blocks behave just as individual cells would in a CA.

For memory $T=2$, one takes 3x3 blocks, and constructs a CA neighborhood of 3x3 blocks.



The output 3x3 block is completely determined if you specify the three 3x3 input blocks above it.

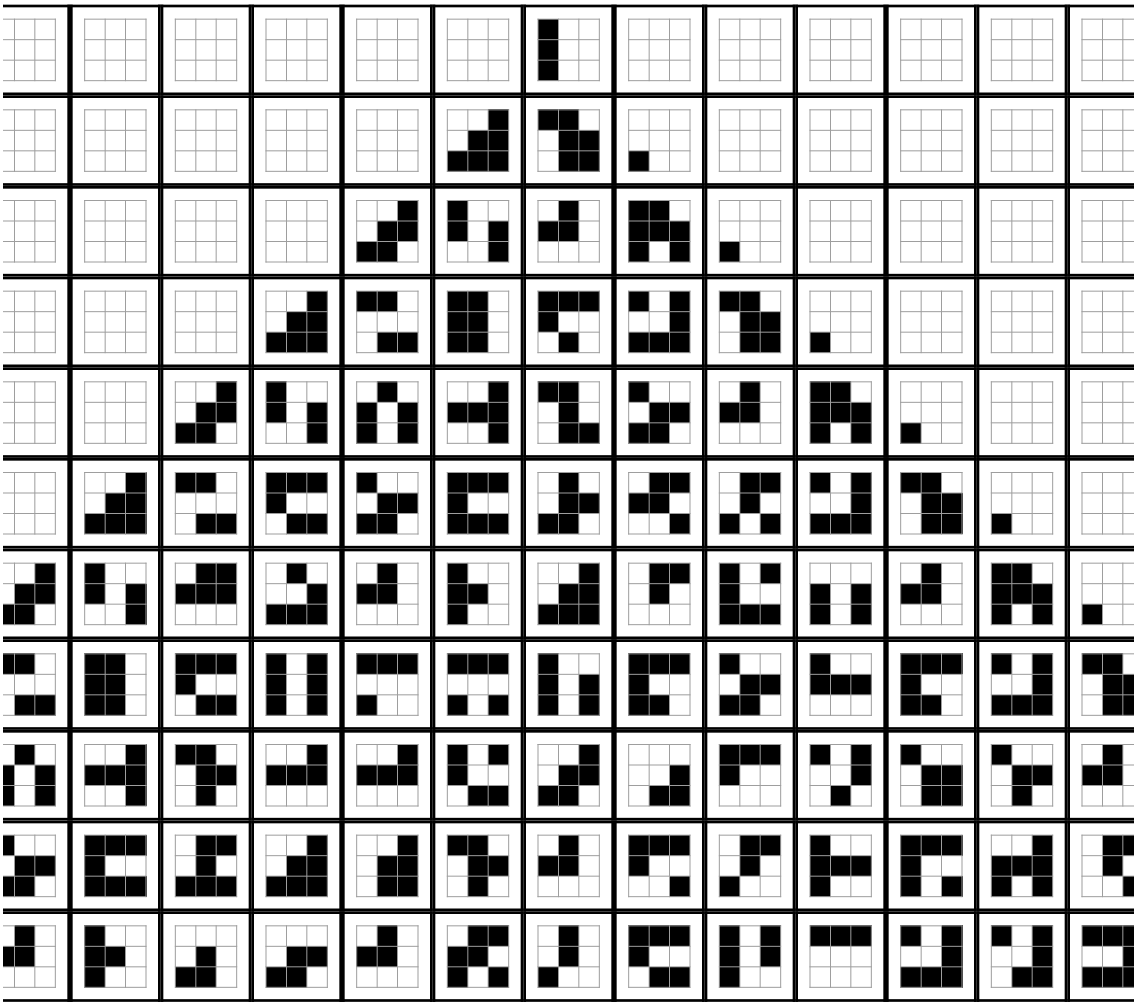
Mapping each 3x3 block to a different color, and then constructing the rule table, one gets a CA with $2^9 = 512$ colors.

In general, a *Block Emulation* of an ECAM-T is a CA with $k = 2^{(T+1)^2}$ colors.

Rule 30 ECAM-2 Block Evolution

Here is rule 30 ECAM-2, partitioned into nonoverlapping 3x3 blocks.

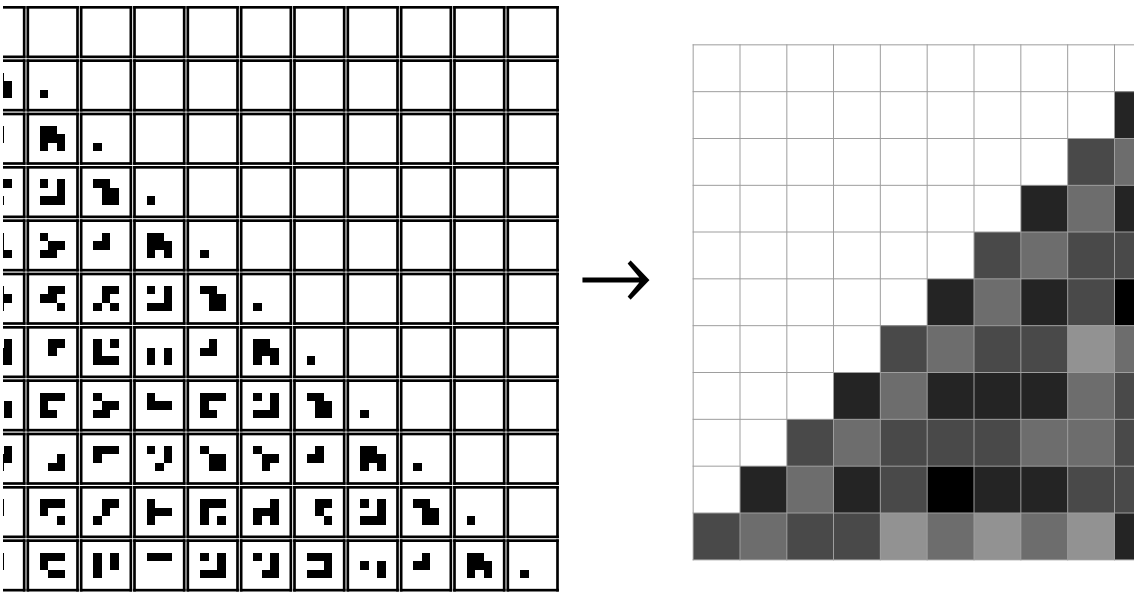
I call this the "Block Representation".



ECAM: Block Density Representation

A simple way to view the block emulation is to plot the average number of black cells in each block (the density).

I call this the "Block Density Representation".



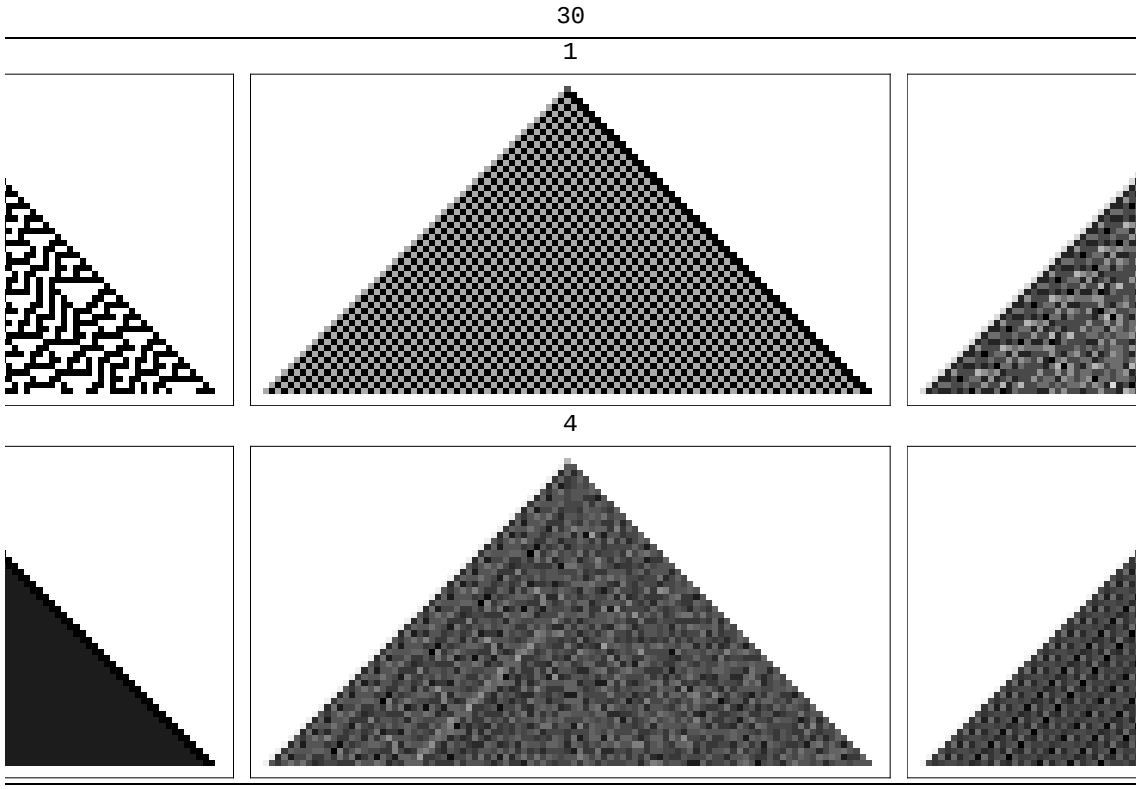
Each cell on the right is actually a 3x3 block of cells.

Note that this is a "lossy" mapping, in that we've given all the blocks with the same density the same color.
Since we don't preserve the identity of each block, this mapping is irreversible.

A Second Look at Rule 30 ECAM-T

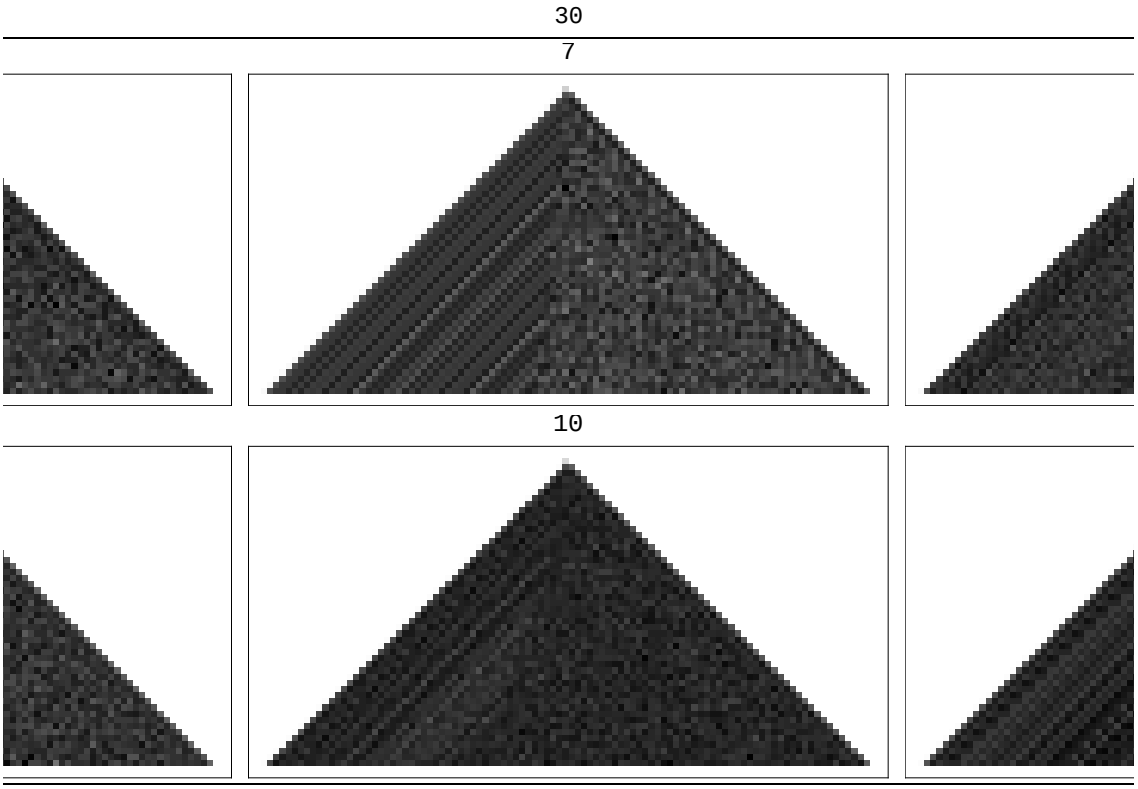
Recall that previously we looked at rule 30 ECAM-*T* for several values of *T*.

Here is the same plot, this time in the block density representation.



Rule 30 ECAM-T

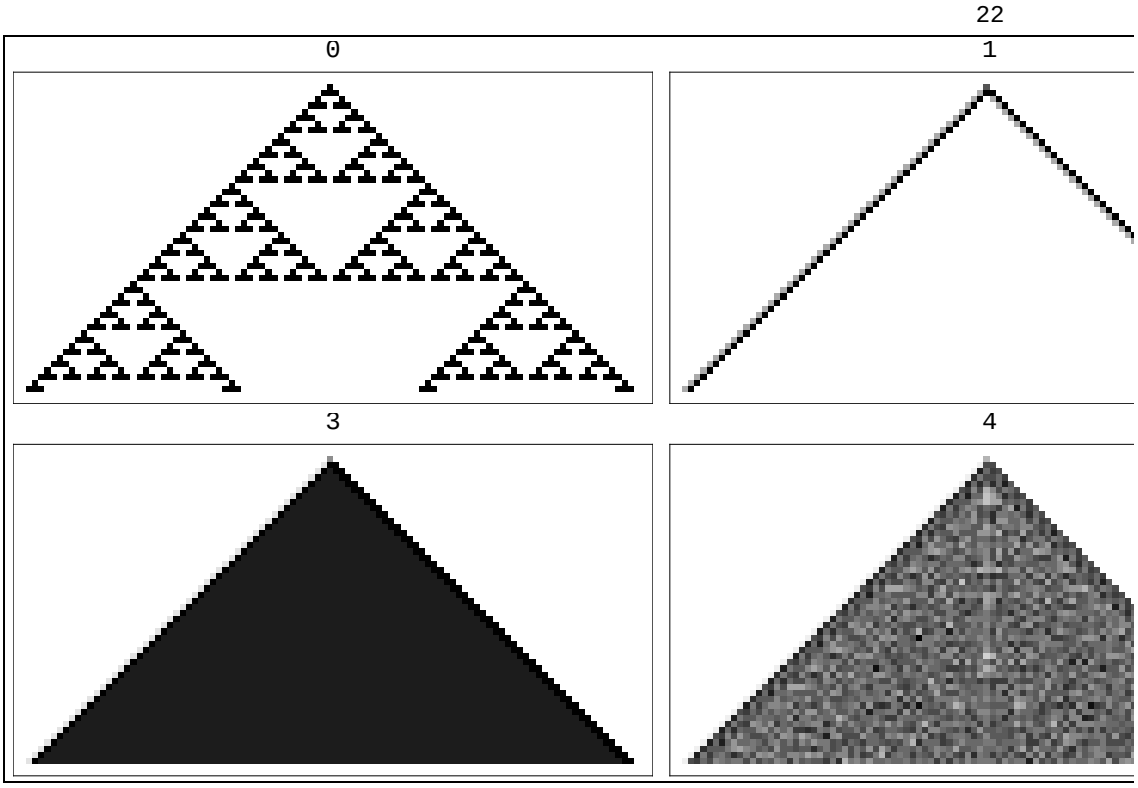
... and with more memory.



It's immediately clear that there are similar features between rule 30 and rule 30 ECAM-*T*, for many values of *T*.

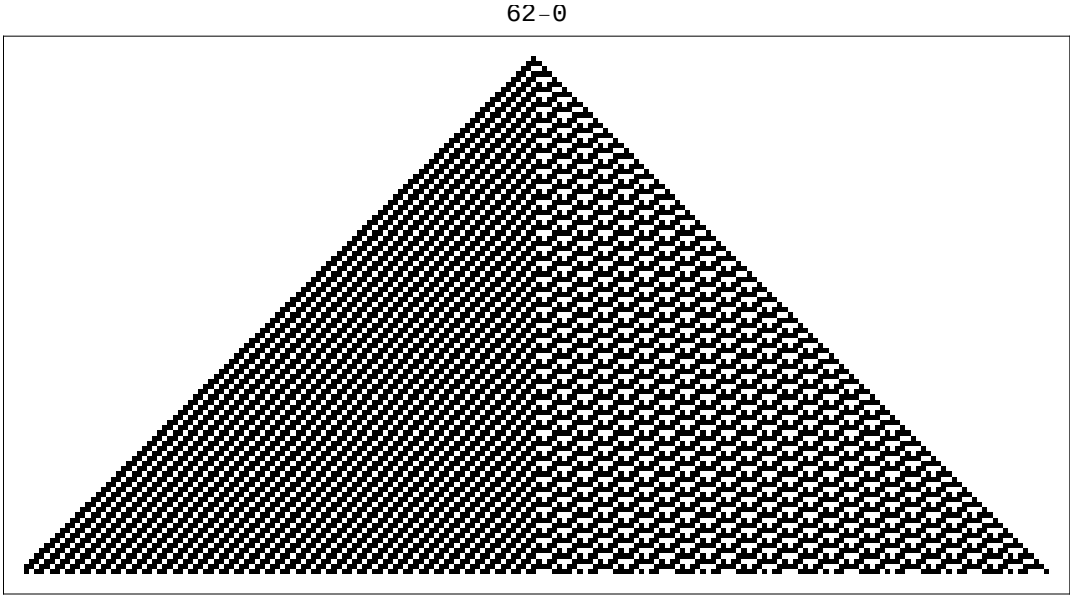
Rule 22 ECAM-T

Here is rule 22 ECAM-*T* in the block density representation. Again, for many *T*, the rule has similar features.

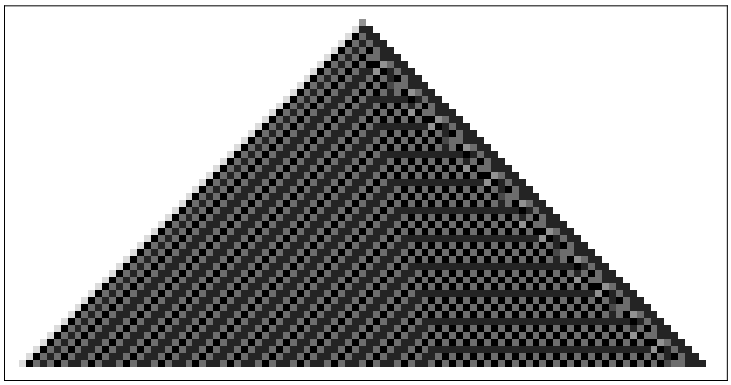


Rule 62 ECAM-T

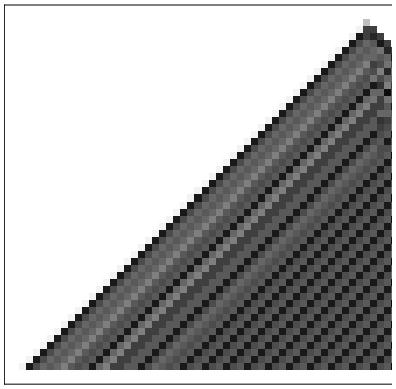
Here is rule 62. Just like for rule 30 and rule 22, there are details that are different, but the overall form is similar.



62:2

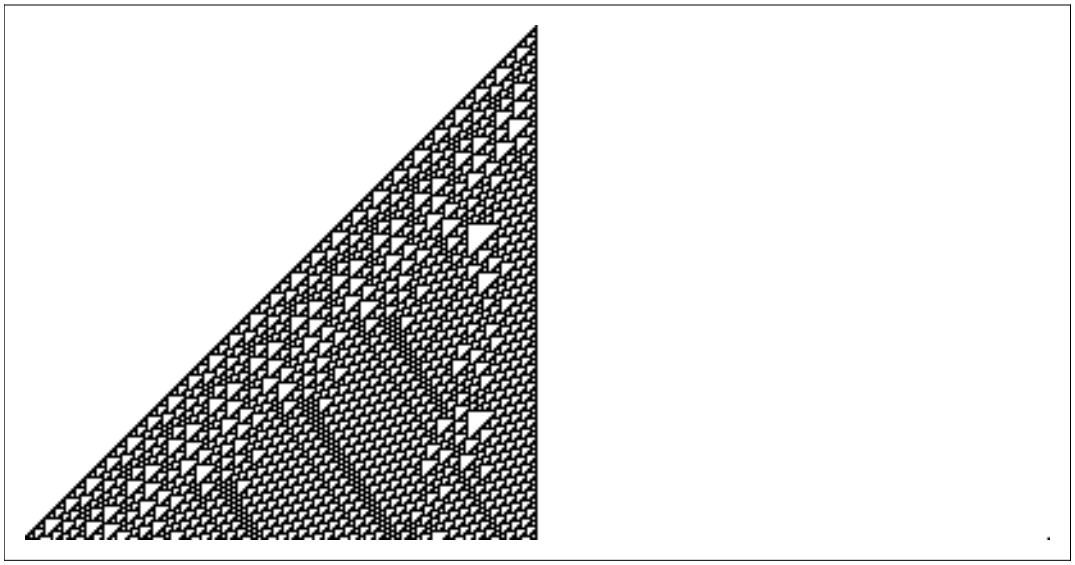


62:4

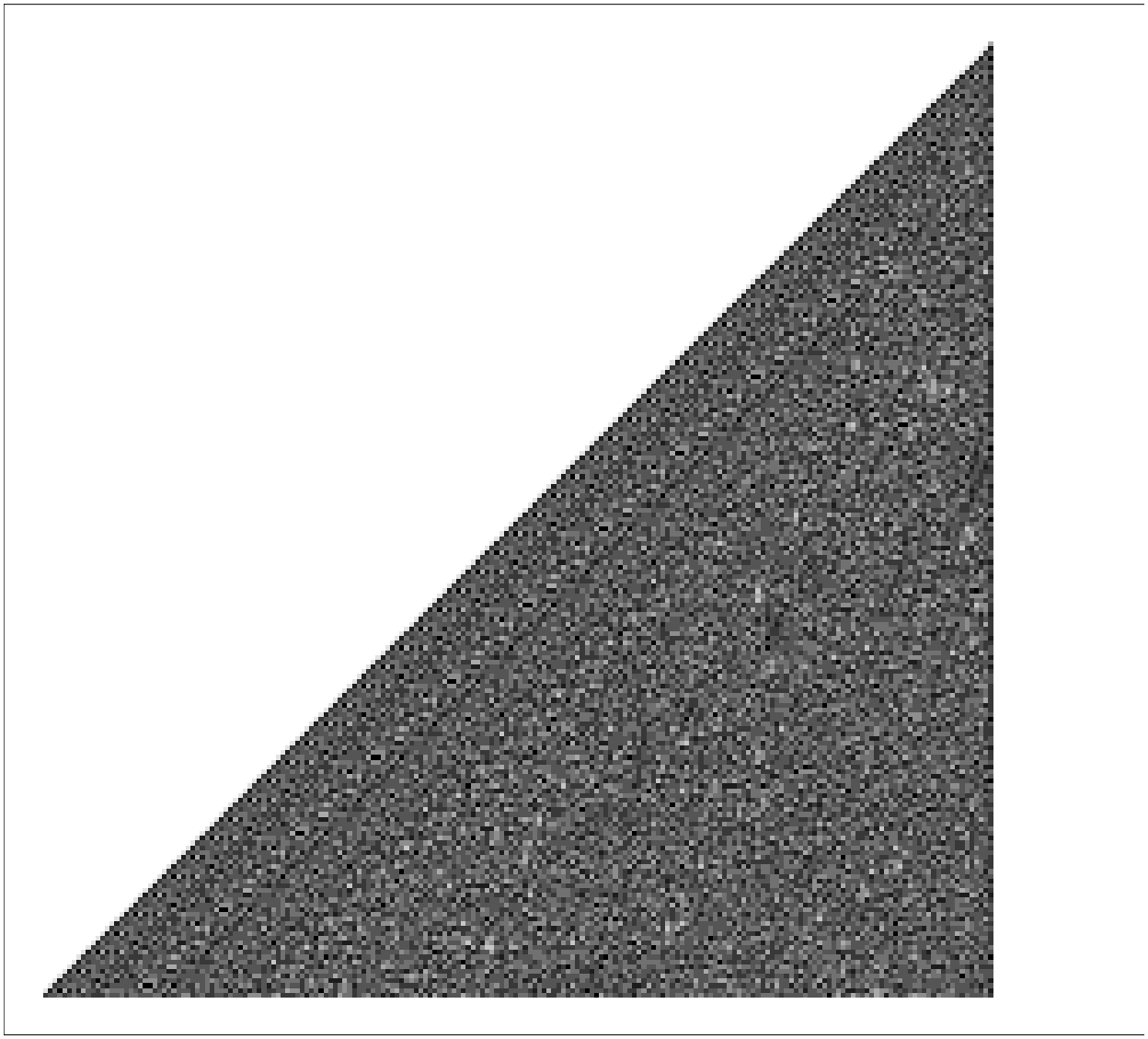


Rule 110 ECAM-2

"110-0"



"110:2"

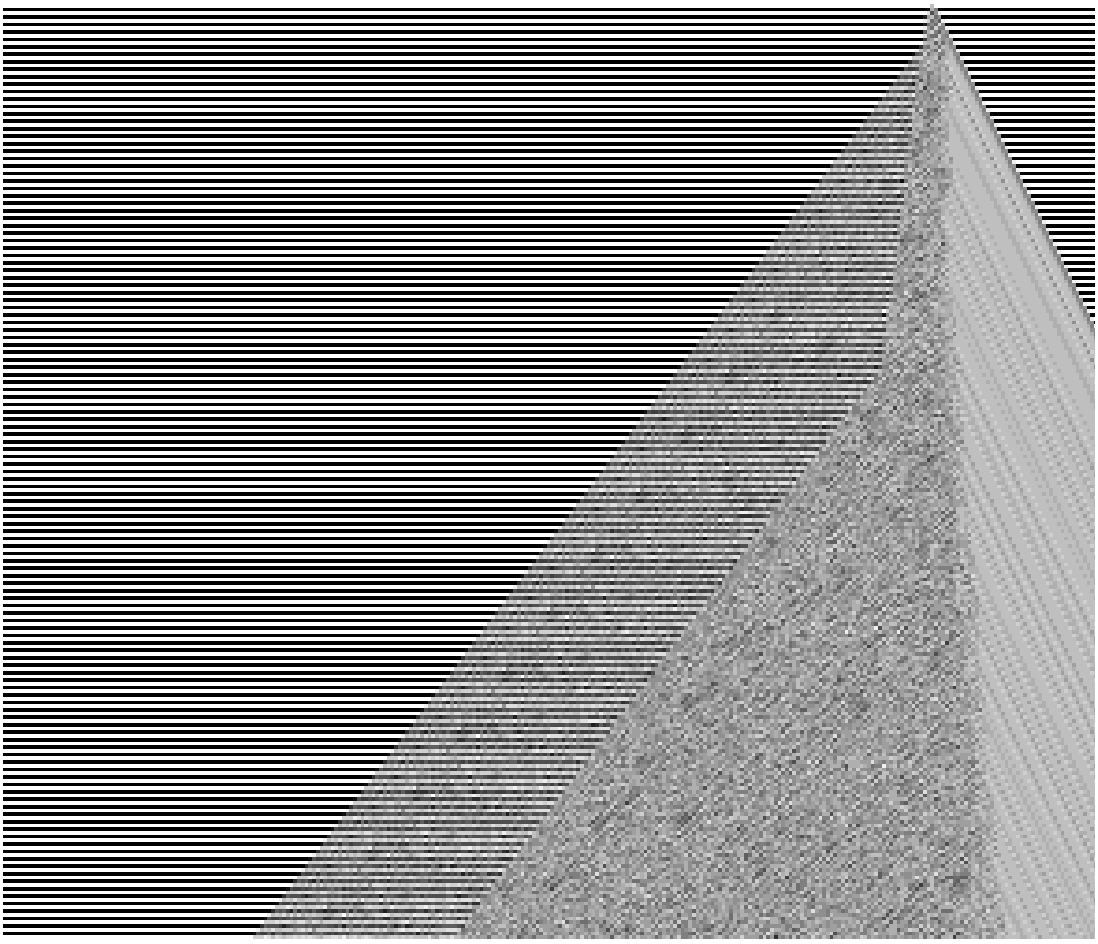


Interesting Growth Rule 41 ECAM-3

This is rule 41 ECAM-3 from one of the 65 thousand single 4x4 block initial conditions, shown in the block-density representation.

Notice the sudden change in the growth rate of the inner structure. Is this a phase transition?

41:3



A Robust Phenomenon

Across all ECAM, the block representation clearly shows that there is a fundamental similarity between rule x and rule x ECAM-T.

Conclusions

ECAM are Elementary Cellular Automata with Memory, where the centre cell is referenced from a number of time steps in the past.

We looked exhaustively at all ECAM-1 and ECAM-2 from simple and random initial conditions. We found that ECAM-1 only do repetitive or nested things from a simple initial condition, and ECAM-2 do more complex things from a simple initial condition.

Any ECAM-*T* (where *T* is the memory of the centre cell) can be emulated by a CA without mem-

ory, and 2^{T+1} colors.

Any ECAM-T also has an emulation that evolves squre blocks of cells of side length $T+1$. Mapping blocks to colors, one gets a CA with $2^{(T+1)^2}$ colors.

In the "Block Density Representation", one plots the density of black cells in each block.

Using the block density representation, a given rule has similar features for most values of T.

Acknowledgements

Todd Rowland: collaborator at Wolfram Research

Stephen Wolfram: original idea for memory scheme (2004)

Karl-Peter Marzlin: supervisor, Institute for Quantum Information Science, Department of Physics, University of Calgary

Christian Jacob: supervisor, Artificial Intelligence Laboratory, Department of Computer Science, University of Calgary

Eric Rowland: block emulation idea

National Science and Engineering Research Council of Canada (NSERC): major funding