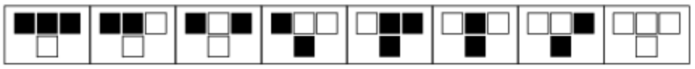
Solutions for Math Circles 2018

November 28

1. Here is Rule 30.

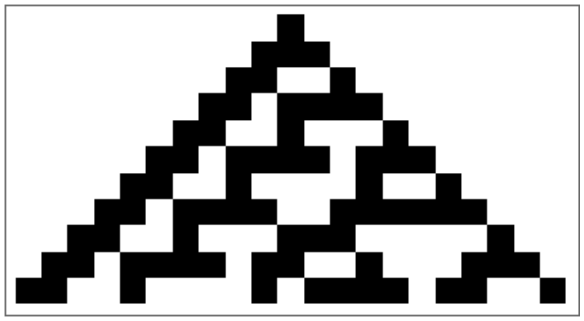


Starting with a single live cell, iterate these rules ten times by hand. Then open up Wolfram Lab and enter the code

```
ArrayPlot[CellularAutomaton[30,{{1},0},10]]
```

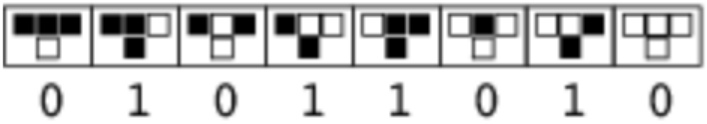
to get the computer to do it.

Solution. Here is the Wolfram Lab output.



2. Use binary expansions to write the transition rules for Rule 90.

Solution. First, convert 90 to binary. You should get 90 = 01011010. So the transition rules are



Notice the order of the rules! It has to be in *this* order if you want to use Wolfram's numbering.

3. Use binary expansions to figure out the rule number for the following transition rules.

Solution. The binary representation you get is 10111110, which is equivalent to 190 in base 10. So this is Rule 190.

4. How many (1-dimensional elementary) cellular automata are there in total?

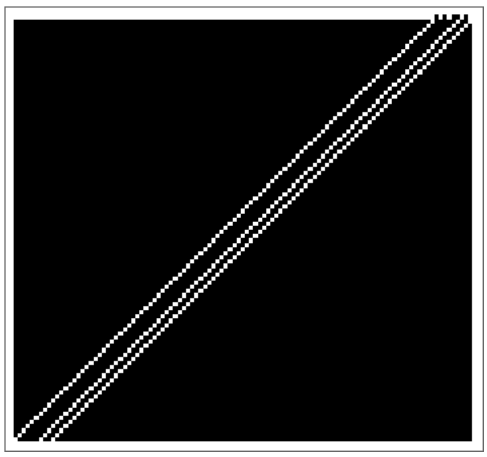
Solution. There are eight rules, and rule can have one of two states (0 or 1). So there are $2^8 = 256$ cellular automata. ∎

5. Enter code into Wolfram Lab to draw Rule 173 with initial state `10101101` (padded with zeros) and 100 iterations. Here `1` means a black cell, and `0` means a white cell.

   Solution. The code you have to enter is

   ```
   ArrayPlot[CellularAutomaton[173,{{1,0,1,0,1,1,0,1},0},100]]
   ```

   The "{{1,0,1,0,1,1,0,1},0}" refers to the block `10101101` padded with zeros. The output you should get is

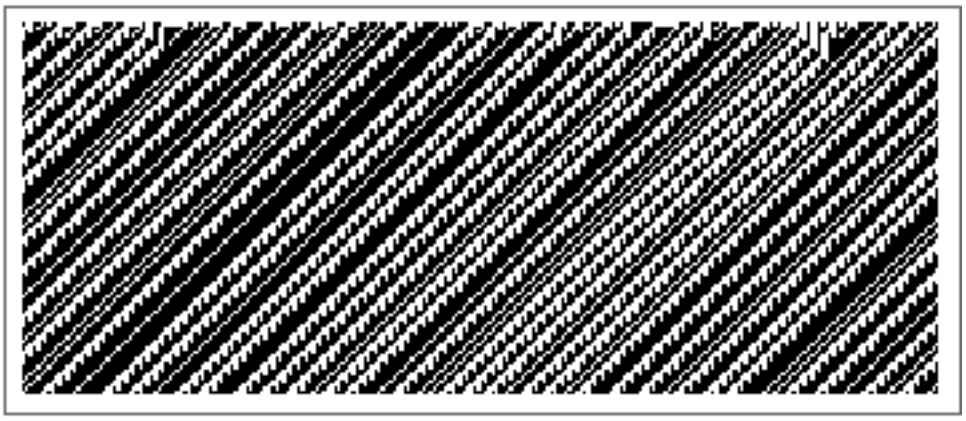
   ∎

6. The code `RandomInteger[1,250]` generates 250 random 0's and 1's. Use this to iterate Rule 173 on some random initial state 100 times.

   Solution. The code you should enter is

   ```
   ArrayPlot[CellularAutomaton[173,RandomInteger[1,250],100]]
   ```

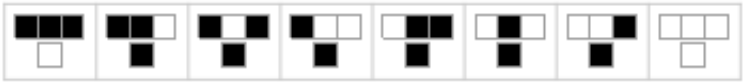
   Your output will probably look different than mine (because it's a random initial state), but here's what I got.

   ∎

7. Enter the code `RulePlot[CellularAutomaton[126]]` to print out the transition rules for Rule 126.

   Solution. Here's what you should get:



   ∎

8. Prove that the state {{1},0} is a Garden Of Eden for Rule 126. (This one is pretty hard: you have to show that Rule 26 will never reach this state no matter what initial state you start with.)

   Solution. Suppose that Rule 126 results in the state $\cdots 0001000 \cdots$. As you can see from the rules in Problem 7, the single `1` can arise in several ways: the states `110`, `101`, `100`, `010`, `011`, and `001` all result in a `1`. Let's consider each of these cases.

   - `110`: Let's suppose you have the following set up.

     ```
     _  1  1  0  _
     ```

     What's the next possible state? Notice that no matter what the _ is on the left, you will still get a live state below the zero.

     ```
     _  1  1  0  _
     _  _  1  1  _
     ```

     We didn't get `010`! So this case is impossible.
   - `101`: Here's our setup.

     ```
     _  1  0  1  _
     ```

     Notice that no matter what the _ are, each of the `1`'s will have `1`'s below them. So the next state will look like:

     ```
     _  1  0  1  _
     _  1  1  1  _
     ```

     This, again, is not `010`. So this case is impossible too!
   - `010`: This case is similar to the last one: you have a block like _`010`_, but then the next state will look like _`111`_.
   - `100`: Now our state is as follows.

     ```
     _  1  0  0  _
     ```

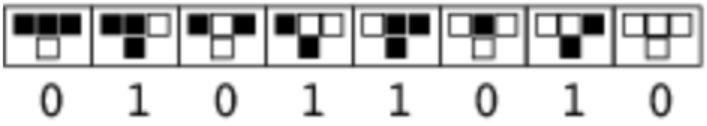
     No matter what the left _ is, the next state will be as follows:

     ```
     _  1  0  0  _
     _  1  1  _  _
     ```

     This, again, is not a single `1` on its own.
   - `011`: This case is symmetric with the case `110`.
   - `001`: This case is symmetric with `100`.

   Phew! We've exhausted every case. This shows that no matter what the previous state is, the next state will NOT be `010`. This shows that `010` has no predecessor, so it's a Garden Of Eden for Rule 126. ∎

9. Here is Rule 90.



Find predecessors for each of the following states in Rule 90.

(a) 00011000

(b) 000111000

(c) 000101100111000

Convince yourself that every state has a predecessor.

Solution. There are actually many different solutions, but I'll give one for each.

(a) $\cdots$ 11110000 $\cdots$

(b) $\cdots$ 0000110100 $\cdots$

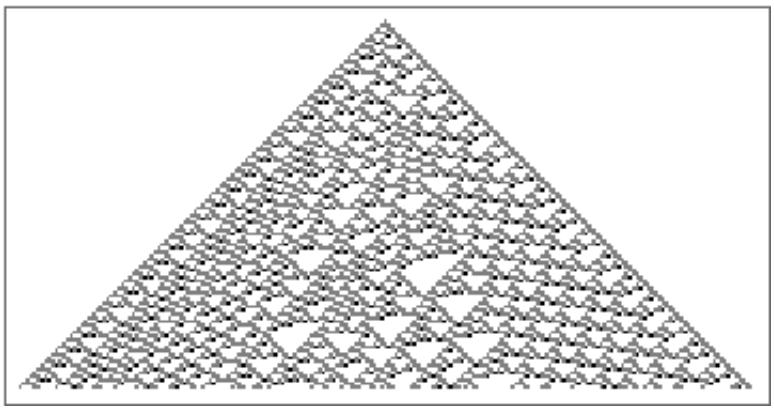
(c) $\cdots$ 11110110100 1111 $\cdots$ ∎

10. **Multistate automata:** The following code will generate the 3-color Rule 679,458

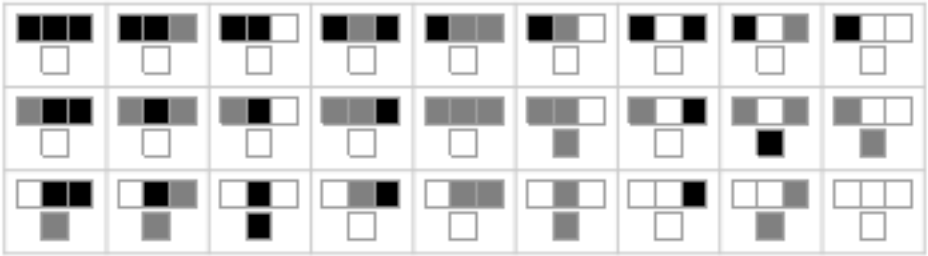
```
CellularAutomaton[{679458,3}]
```

Enter this into Wolfram Lab and see what happens! (Don't forget to encase this line in `ArrayPlot[...]`.)

Solution. I entered the code `ArrayPlot[CellularAutomaton[679458,3,1,0,100]]` into Wolfram Lab, and it gave the following output.



To get the rule plot, just type in `RulePlot[CellularAutomaton[679458,3]]`:

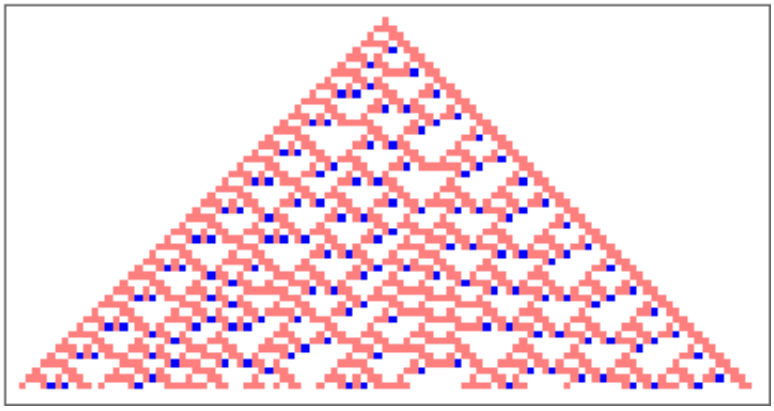


11. To change the colors, use `ColorRule->{1->Pink,2->Blue}`. This will change 1's into pink cells, and 2's into blue cells.

```
ArrayPlot[CellularAutomaton[{679458,3},{{1},0},50],ColorRules->{1->Pink,2->Blue}]
```

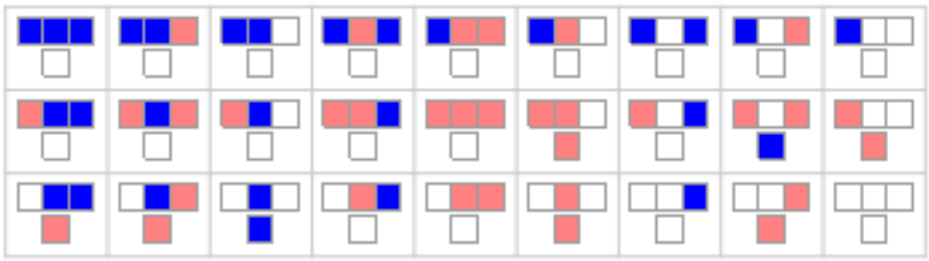
Enter this into Wolfram Lab and see what happens! Also, try to generate the rule plot for 3-color Rule 679,458.

Solution. You should get the following picture:



To get the rule plot, just type in

```
RulePlot[CellularAutomaton[679458,3],ColorRules->{1->Pink,2->Blue}]
```

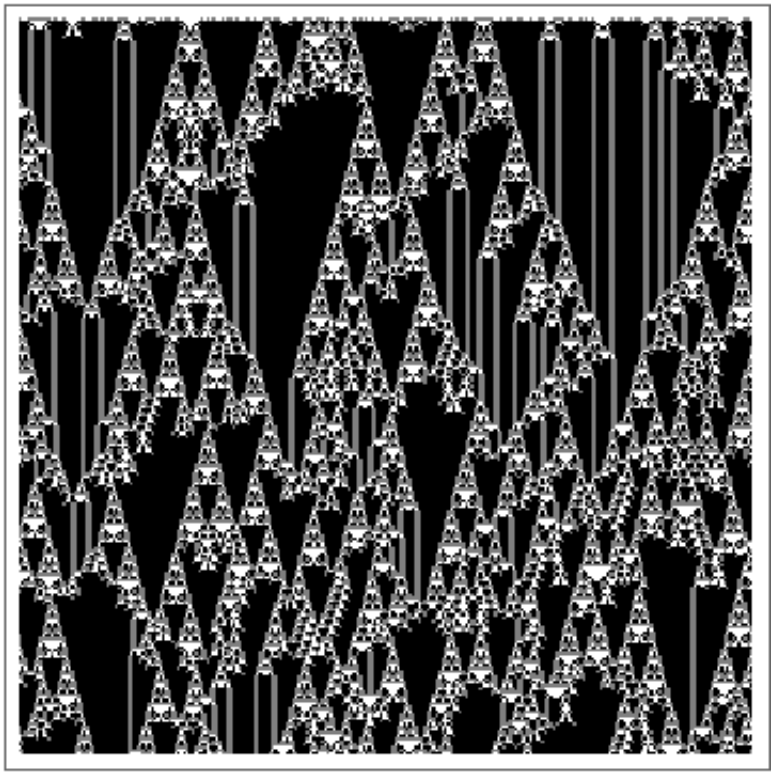


12. **Totalistic rules:** The following code will generate the 3-color totalistic Rule 2049

```
CellularAutomaton[{2049,{3,1}}]
```

(Don't worry about the 1 for now.) Try iterating this on a random initial state, 250 times, and output the plot. Then change the colors: change the 1's to red, and the 2's to orange.

Solution. I used `ArrayPlot` to get the following image.



To change the colors, use `ColorRules->{1->Red,2->Orange}`.