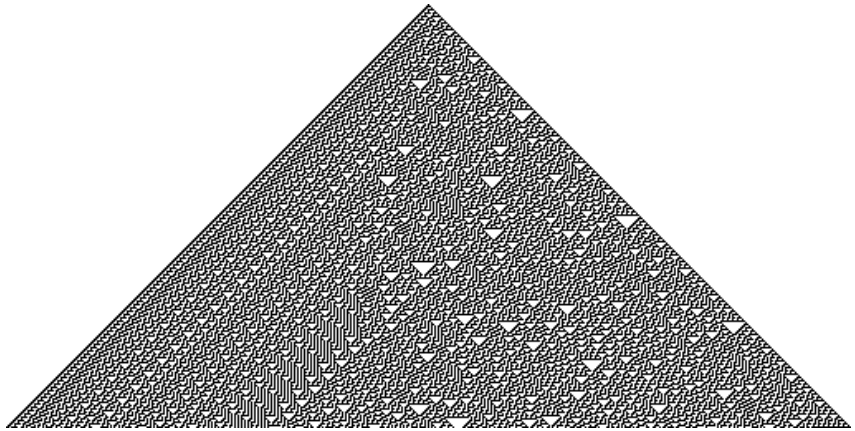
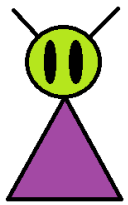
1-Dimensional Cellular Automata



Math Circles 2018
University of Waterloo

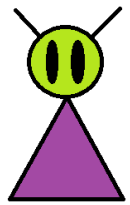
Planet Friendship

This is Zoink. He is a Budling from Planet Friendship.



Planet Friendship

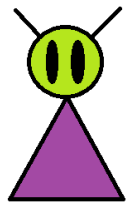
This is Zoink. He is a Budling from Planet Friendship.



At first, Zoink was alone. But Budlings have the special power of **friendification**!

Planet Friendship

This is Zoink. He is a Budling from Planet Friendship.

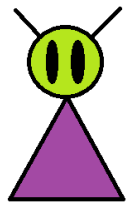


At first, Zoink was alone. But Budlings have the special power of **friendification**!

- Every Budling can create Budlings beside him.

Planet Friendship

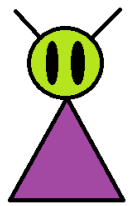
This is Zoink. He is a Budling from Planet Friendship.



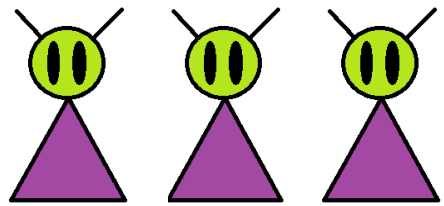
At first, Zoink was alone. But Budlings have the special power of **friendification**!

- Every Budling can create Budlings beside him.
- But if a Budling gets surrounded, he dies!

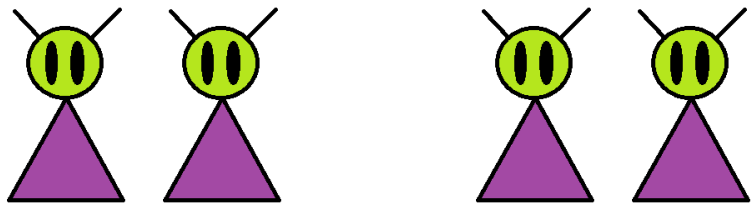
Day 1



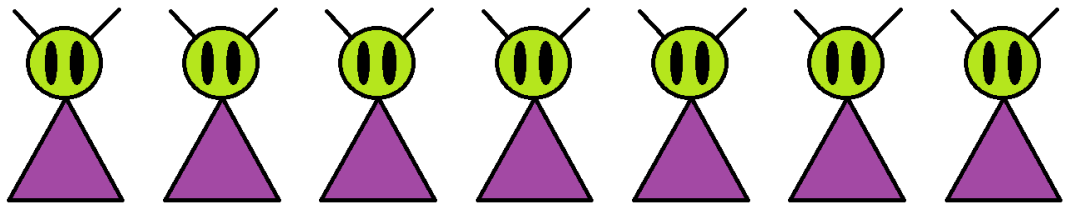
Day 2



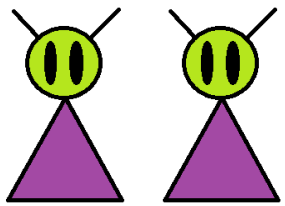
Day 3

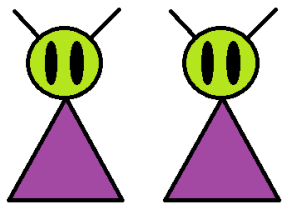


Day 4

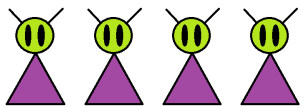


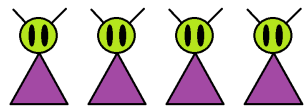
Day 5





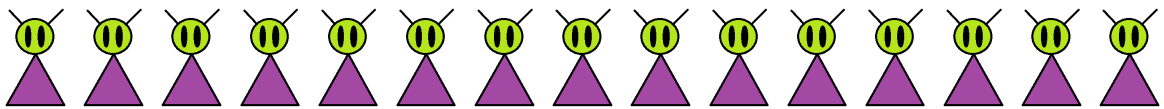
Day 6



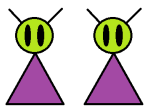


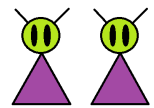
Day 7

Day 8

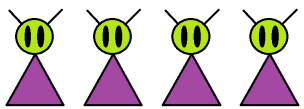


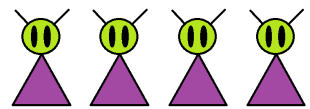
Day 9





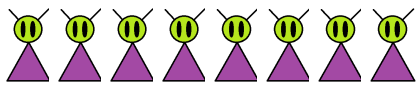
Day 10

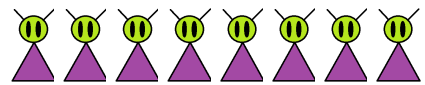




Day 11

Day 12



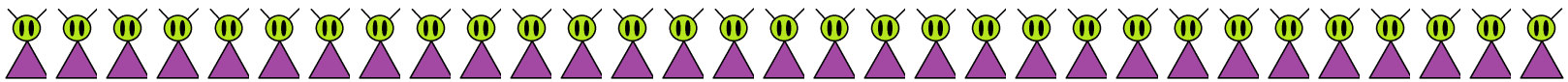


Day 13

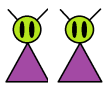
Day 14

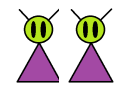
Day 15

Day 16

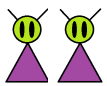


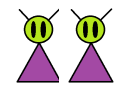
Day 17





Day 17

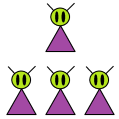




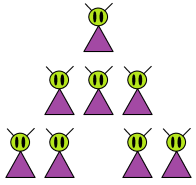
... and so continues the vicious cycle of death on Planet Friendship.

First 16 days on Planet Friendship

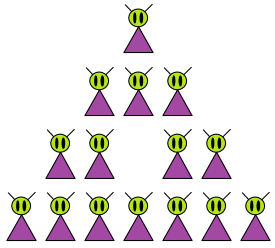
First 16 days on Planet Friendship



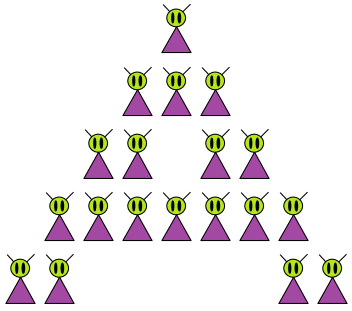
First 16 days on Planet Friendship



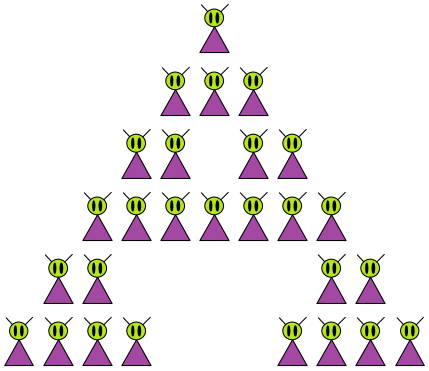
First 16 days on Planet Friendship



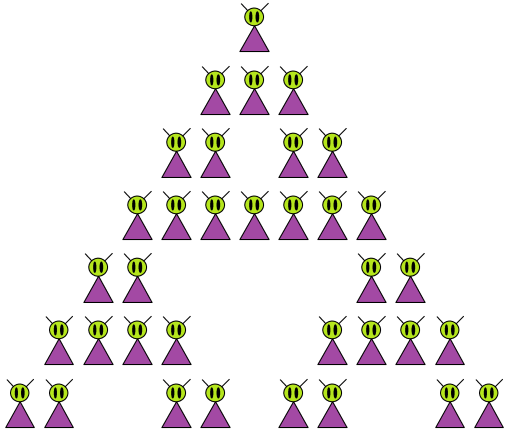
First 16 days on Planet Friendship



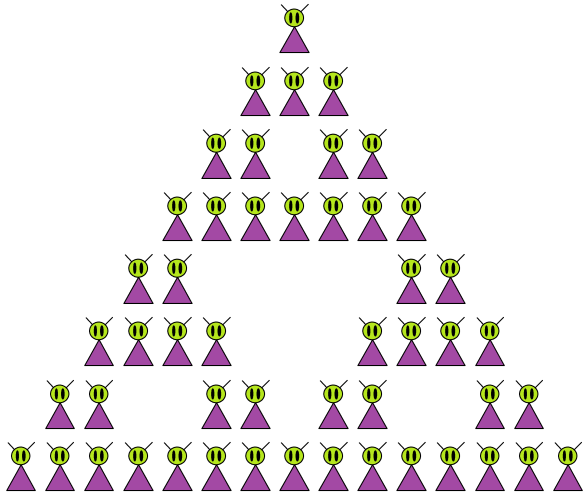
First 16 days on Planet Friendship



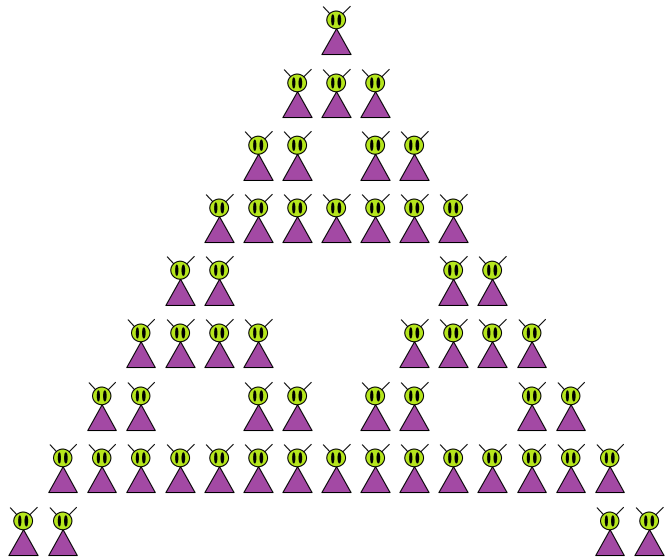
First 16 days on Planet Friendship



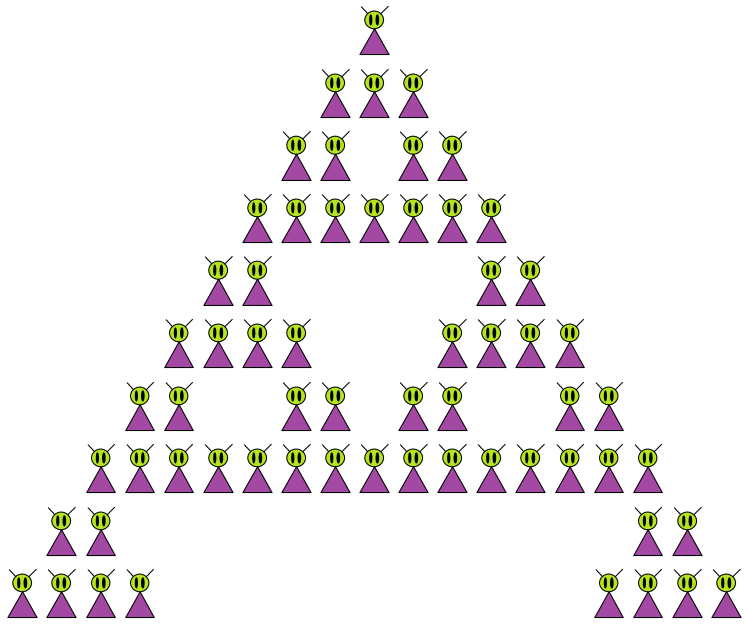
First 16 days on Planet Friendship



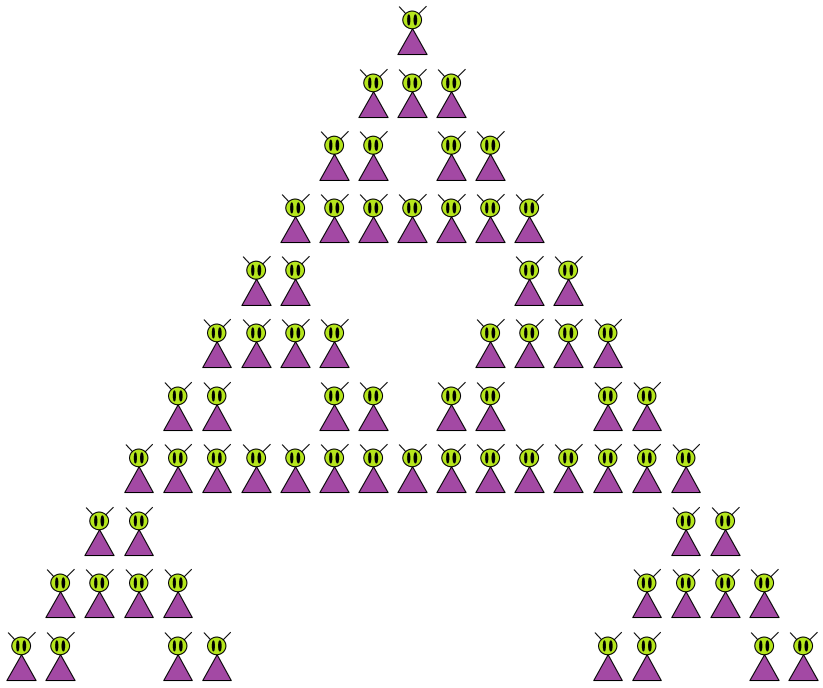
First 16 days on Planet Friendship



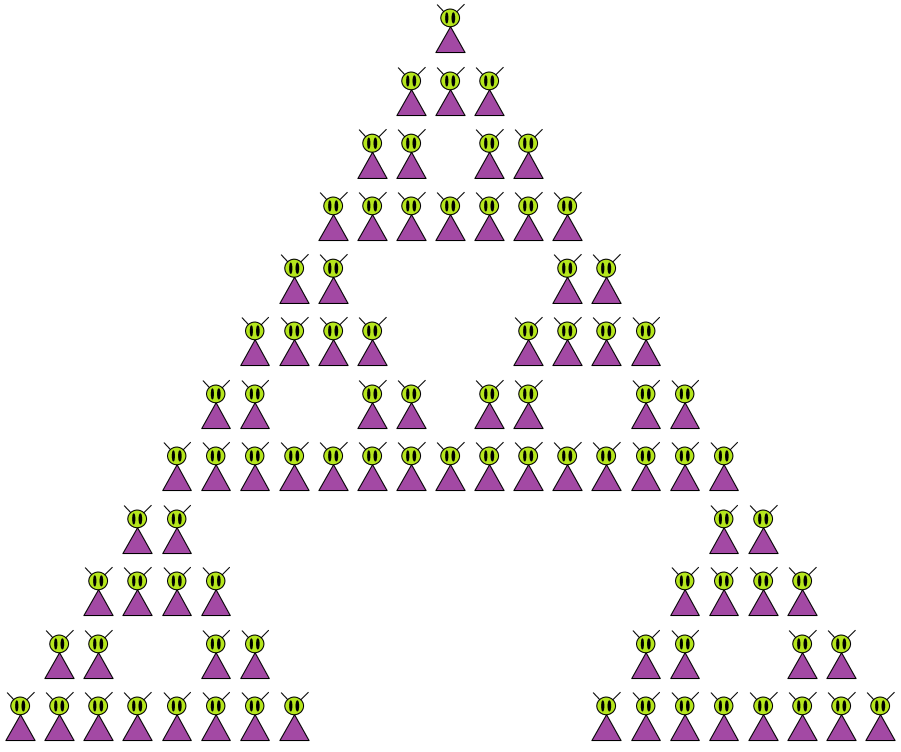
First 16 days on Planet Friendship



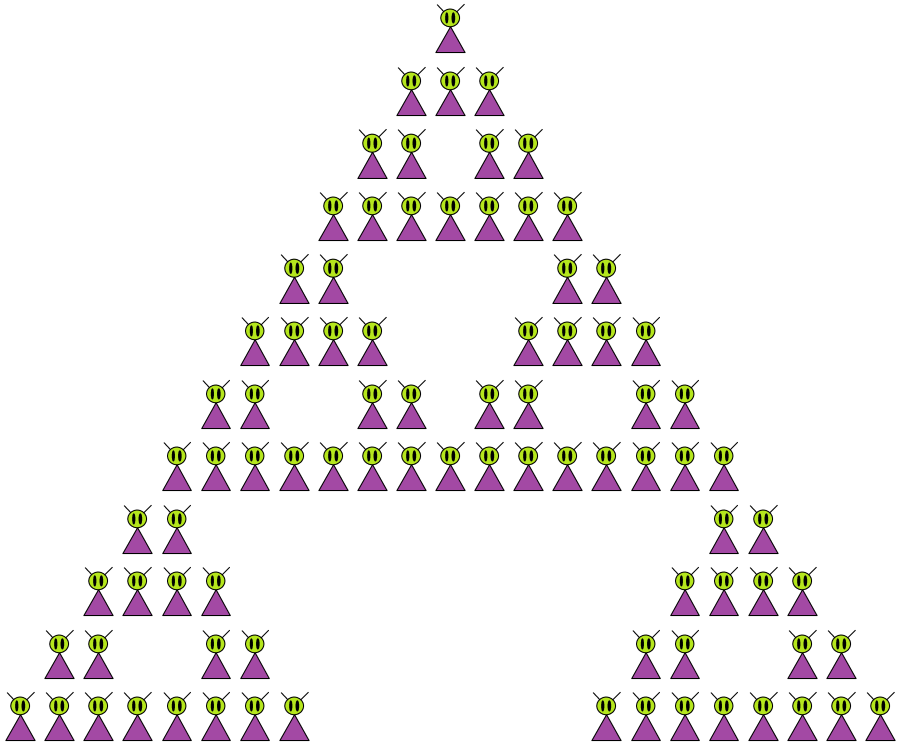
First 16 days on Planet Friendship



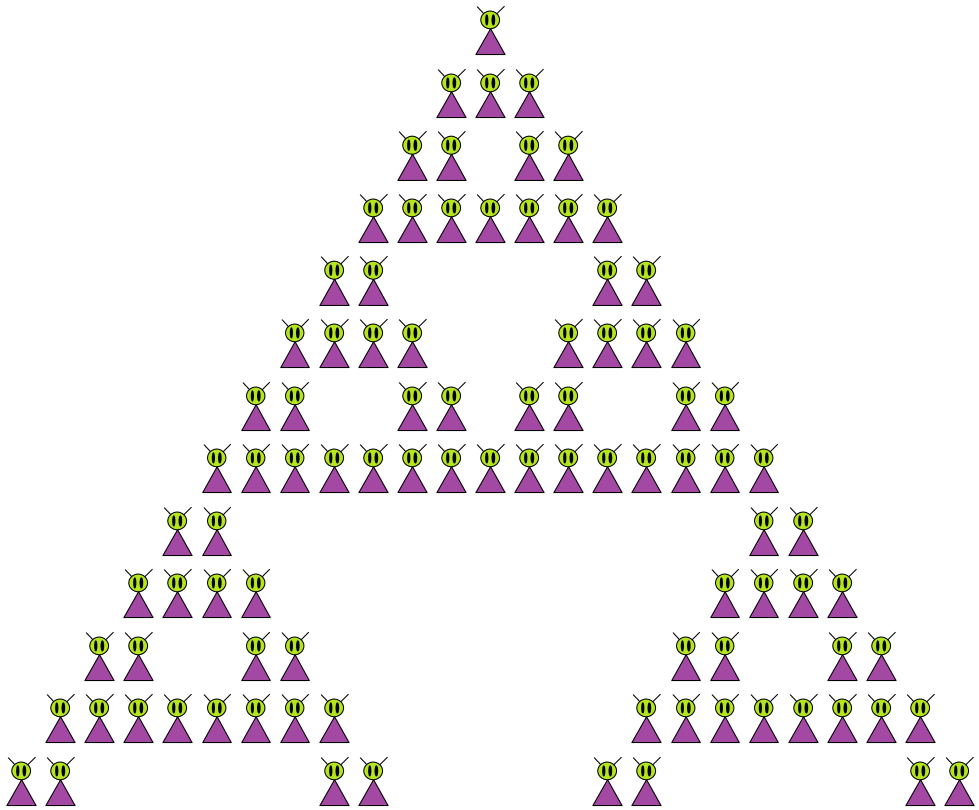
First 16 days on Planet Friendship



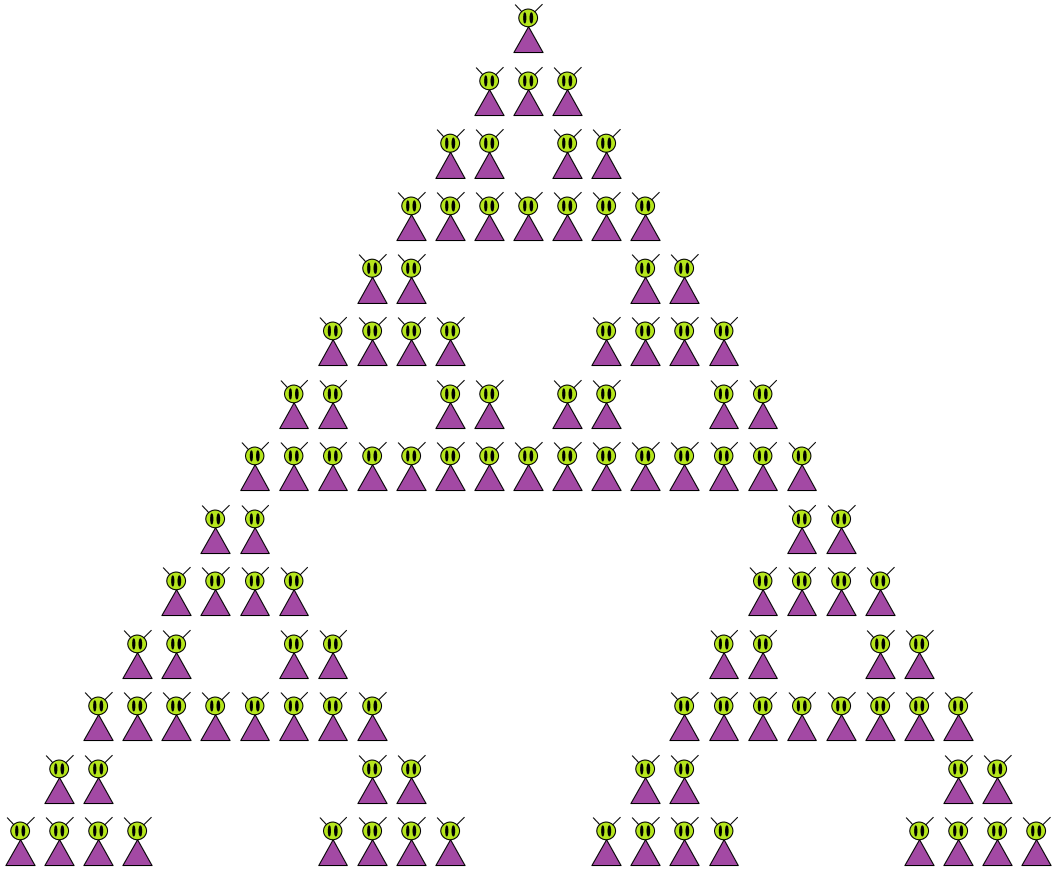
First 16 days on Planet Friendship



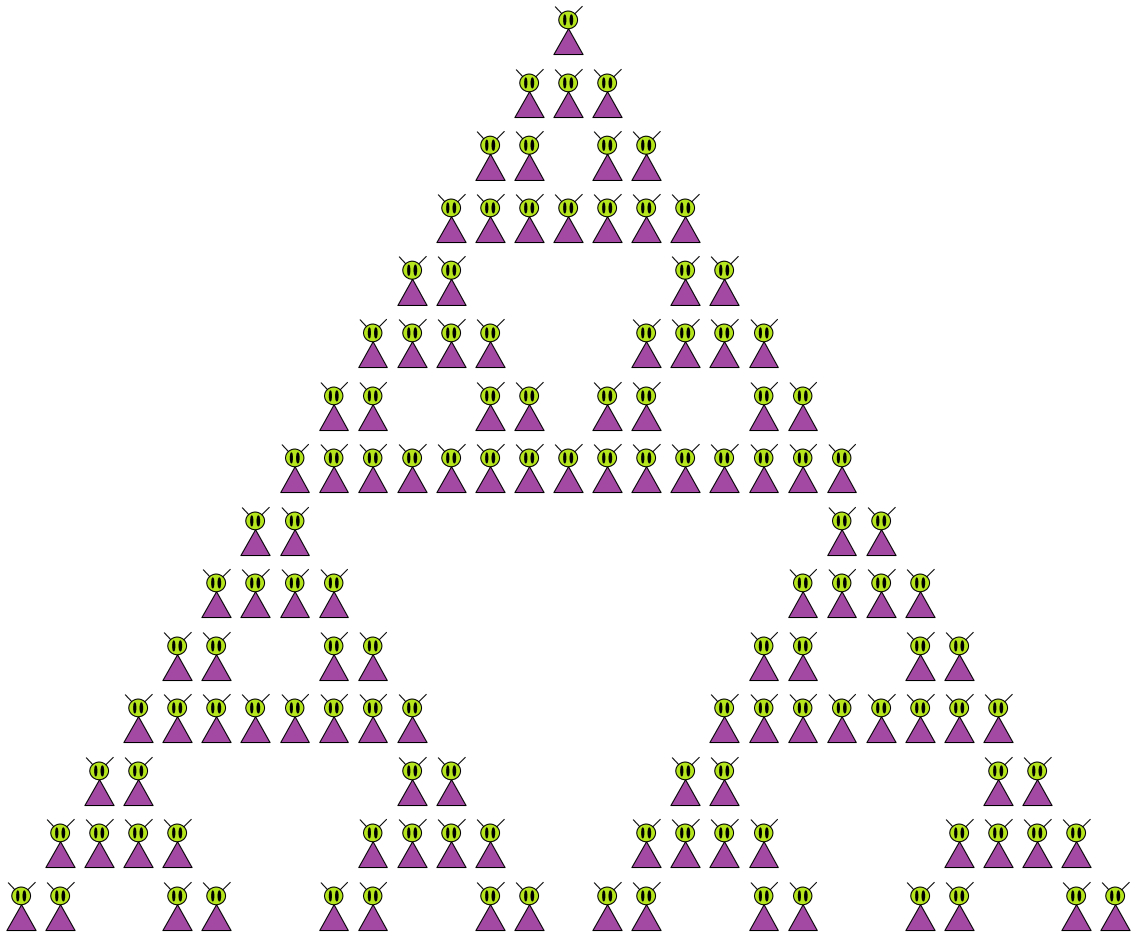
First 16 days on Planet Friendship



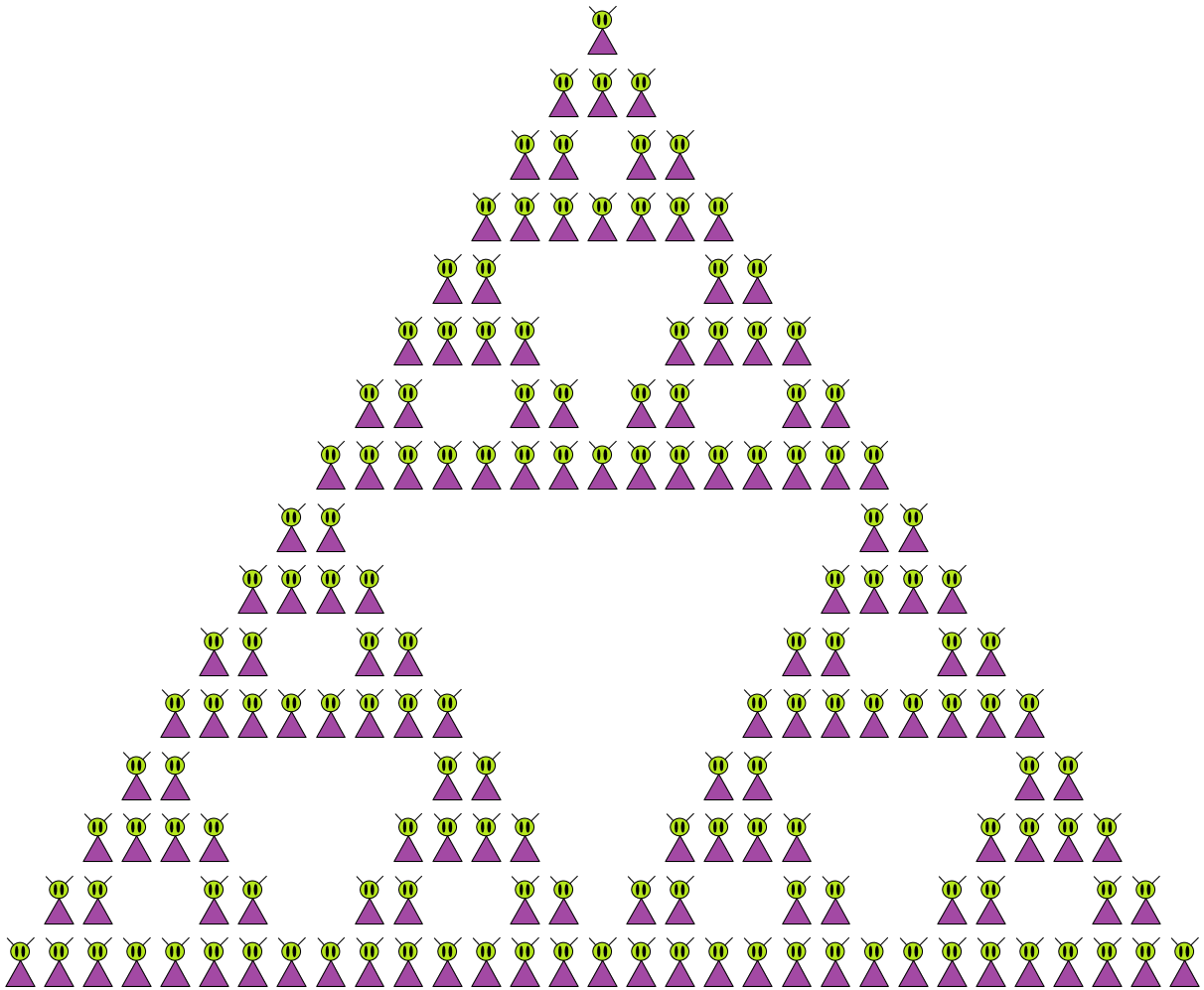
First 16 days on Planet Friendship



First 16 days on Planet Friendship



First 16 days on Planet Friendship

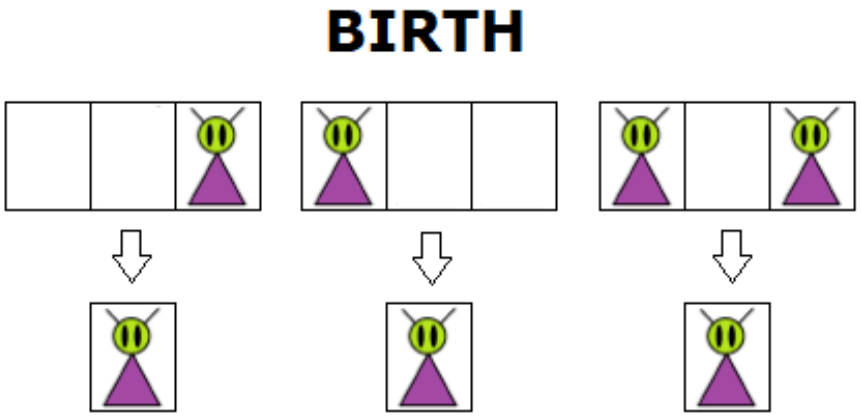


Birth, survival, and death

The rules on Planet Friendship are very precise.

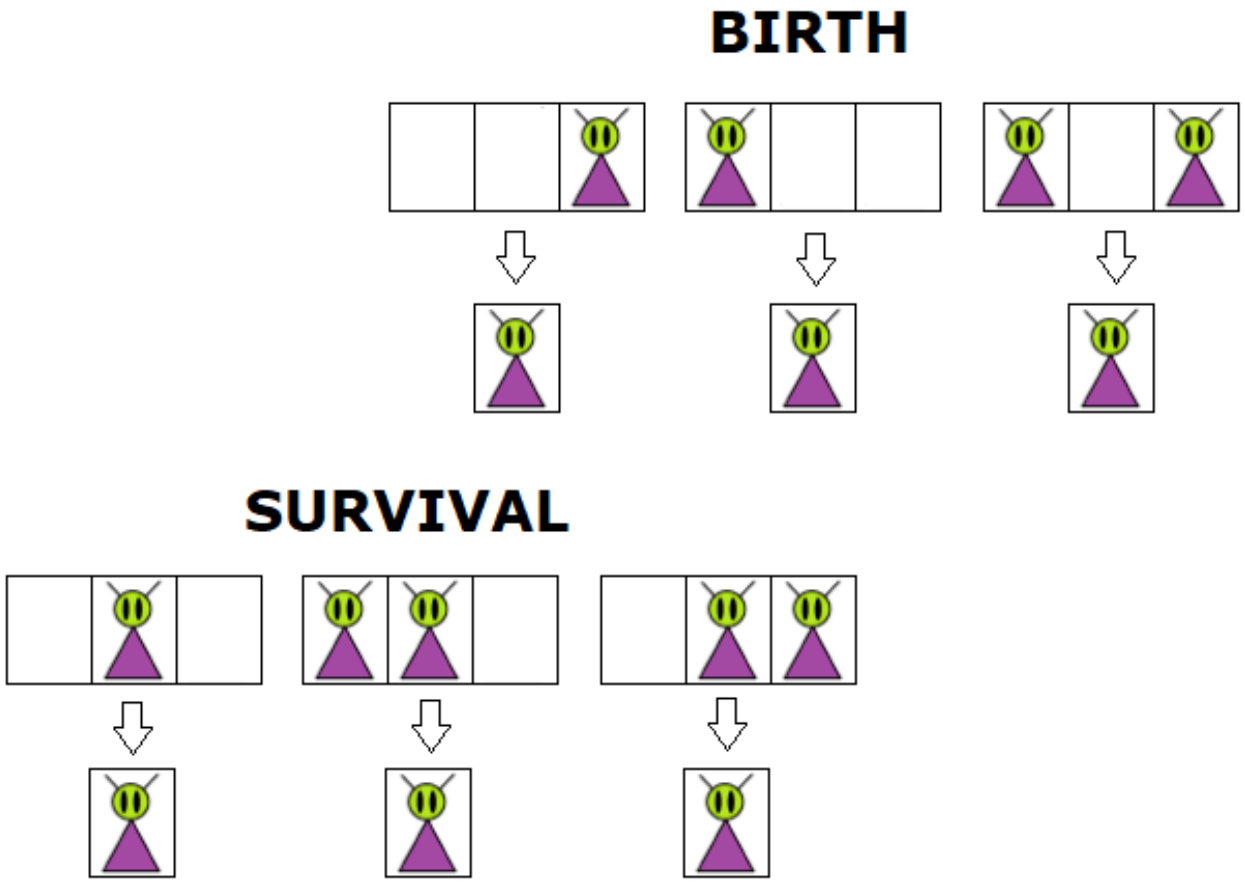
Birth, survival, and death

The rules on Planet Friendship are very precise.



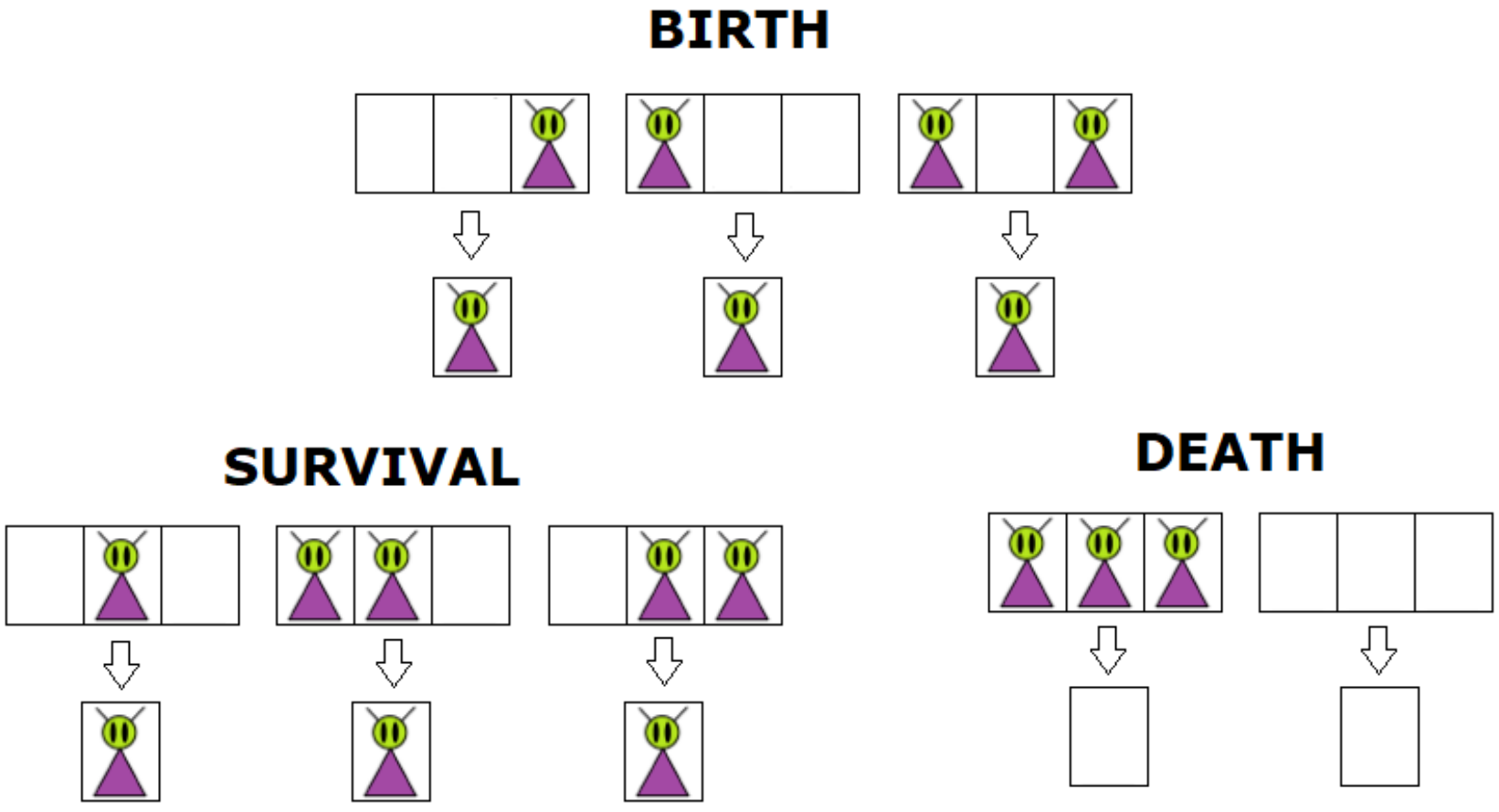
Birth, survival, and death

The rules on Planet Friendship are very precise.

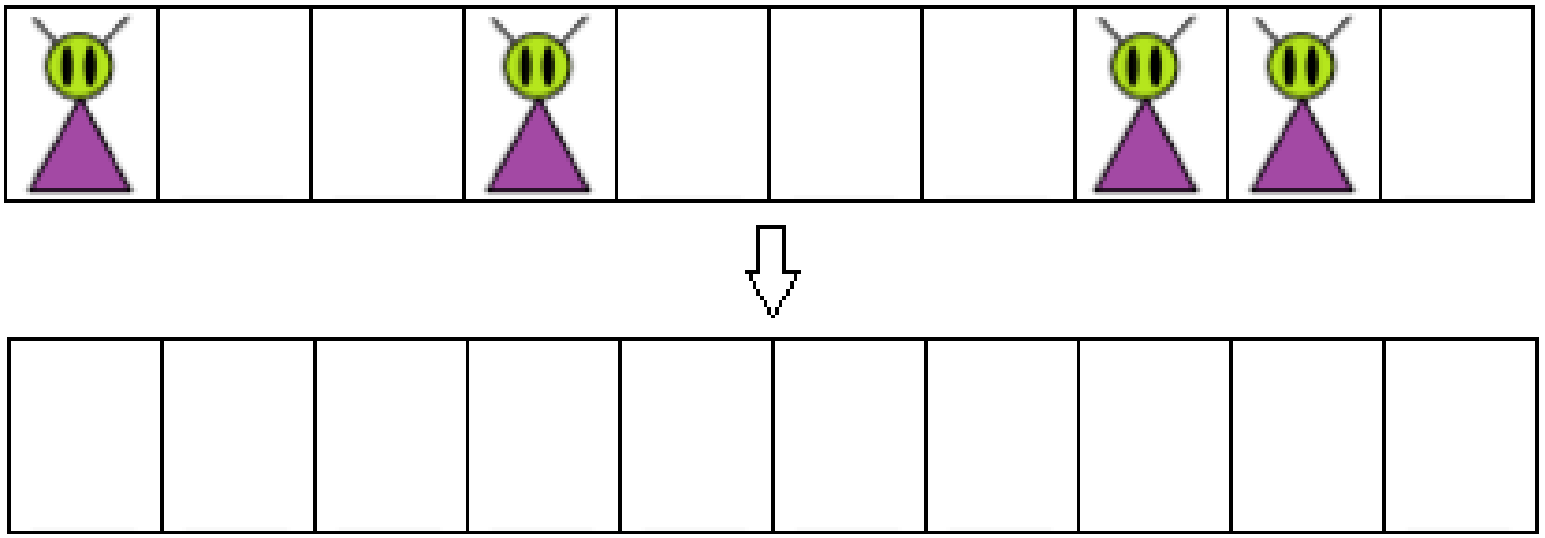


Birth, survival, and death

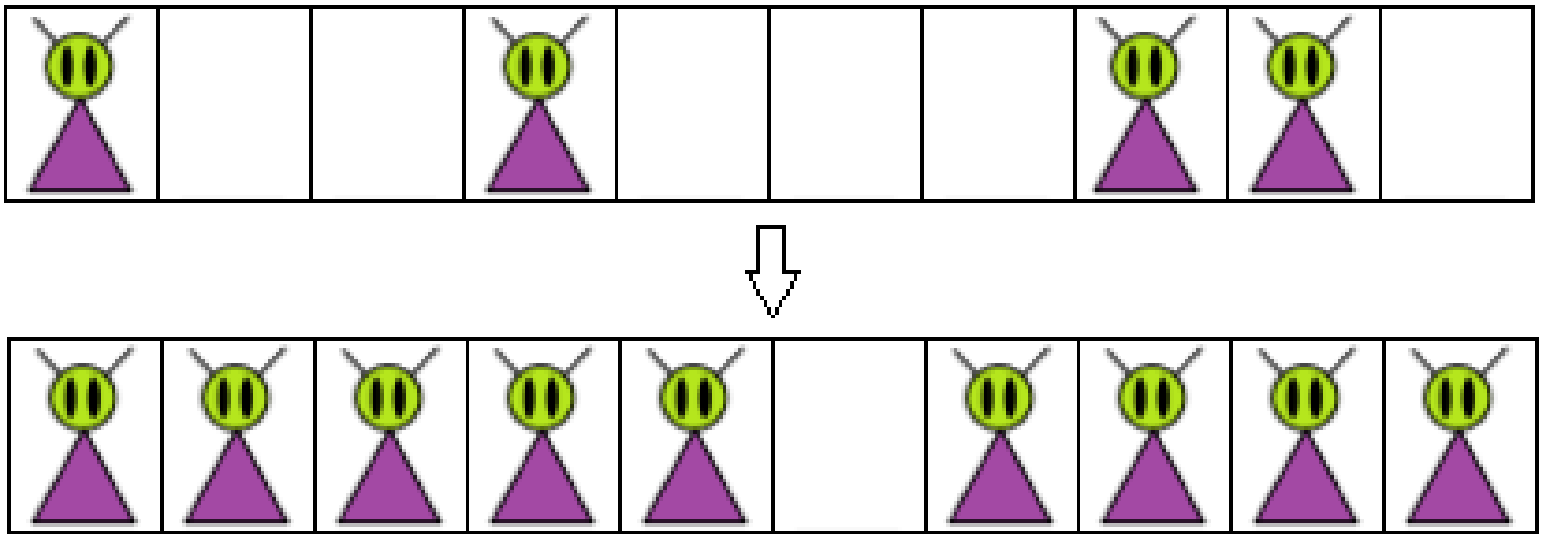
The rules on Planet Friendship are very precise.



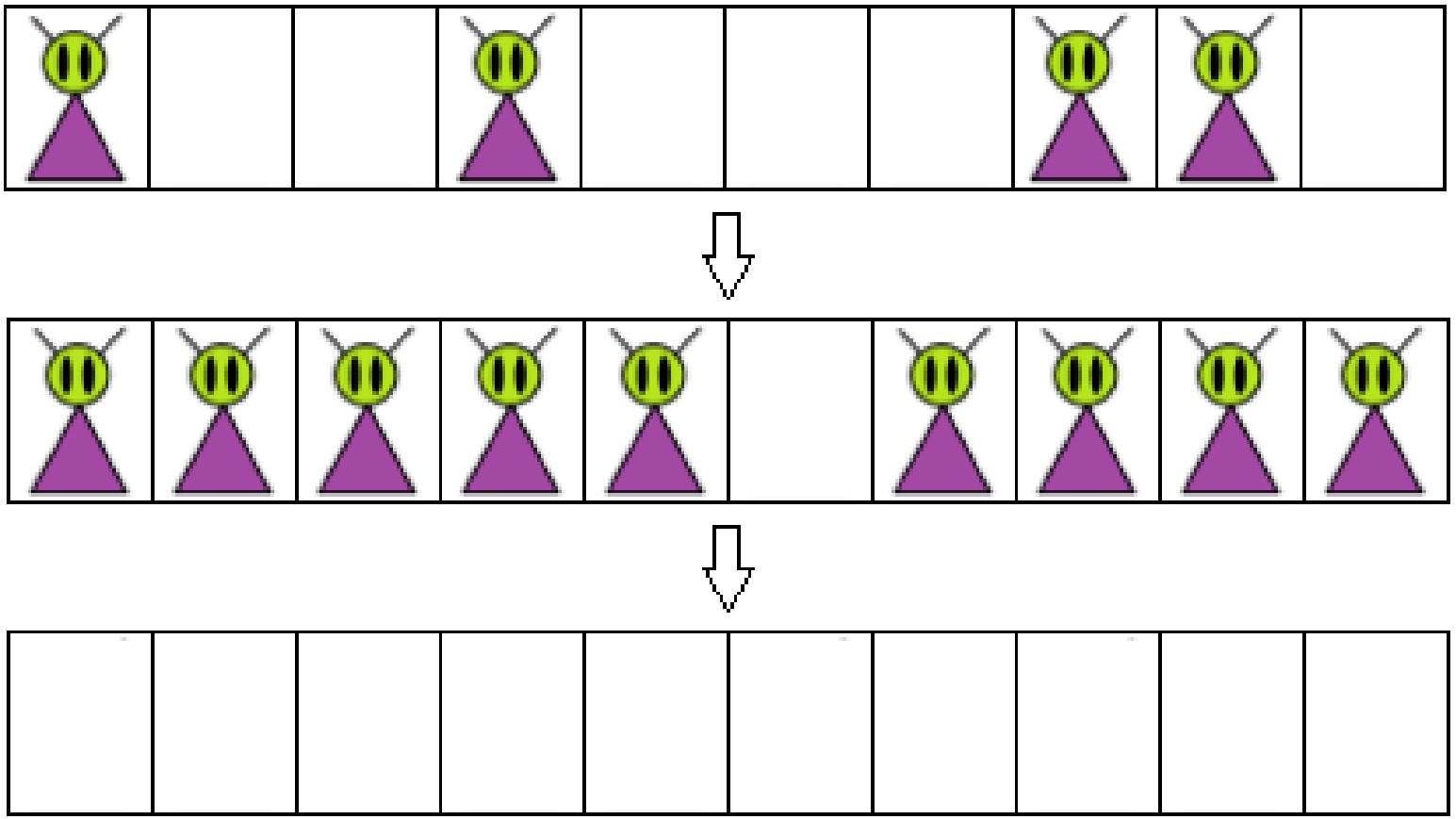
Example



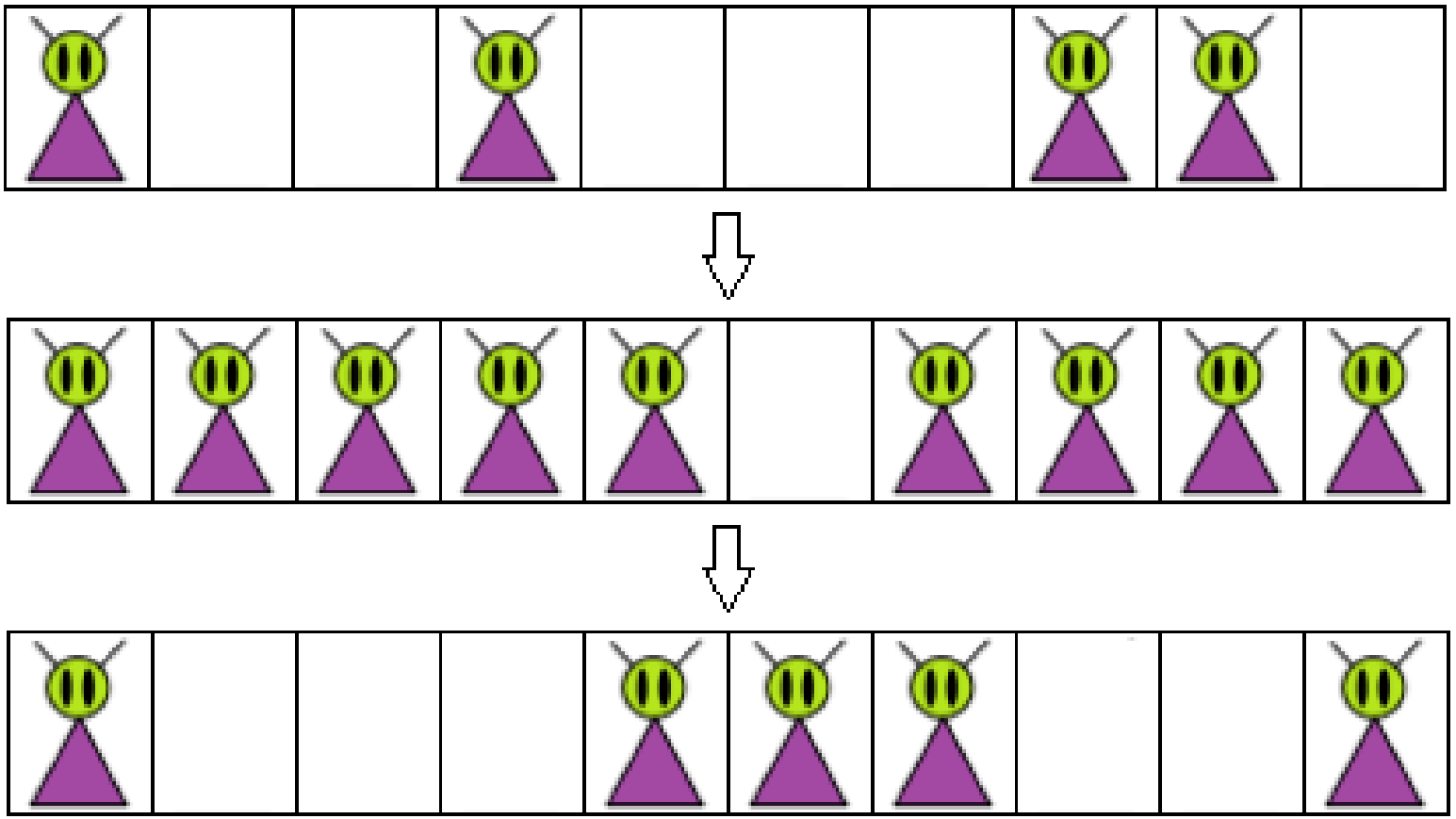
Example



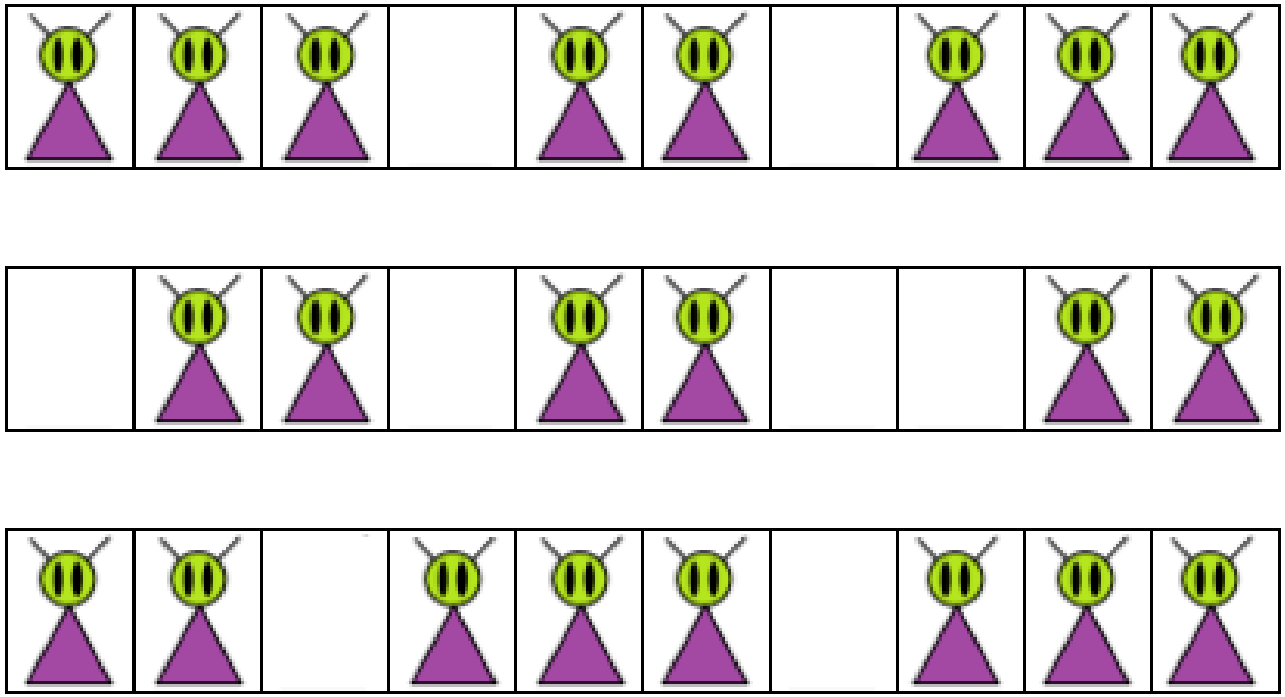
Example



Example

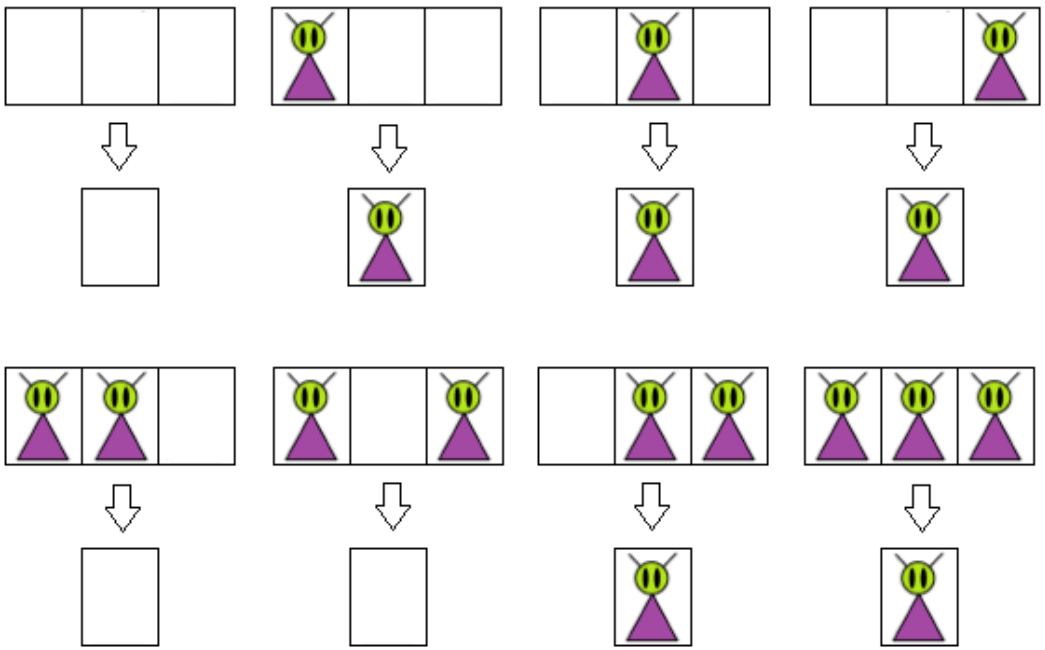


Exercise: According to the rules of friendification, which of the following arrangements of Budlings can occur?

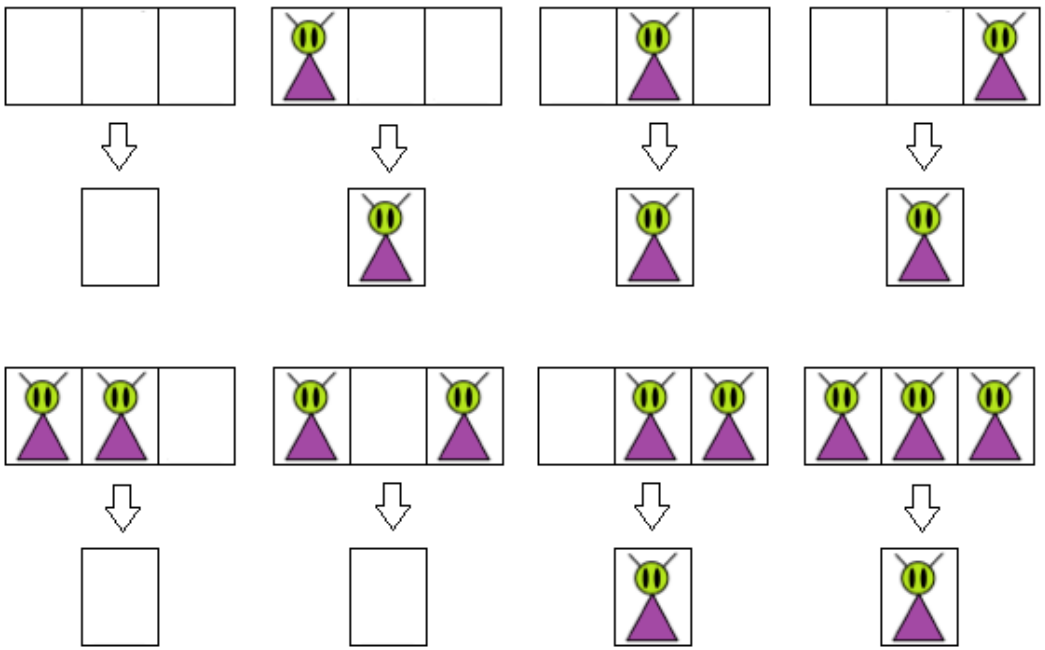


What about different generation rules?

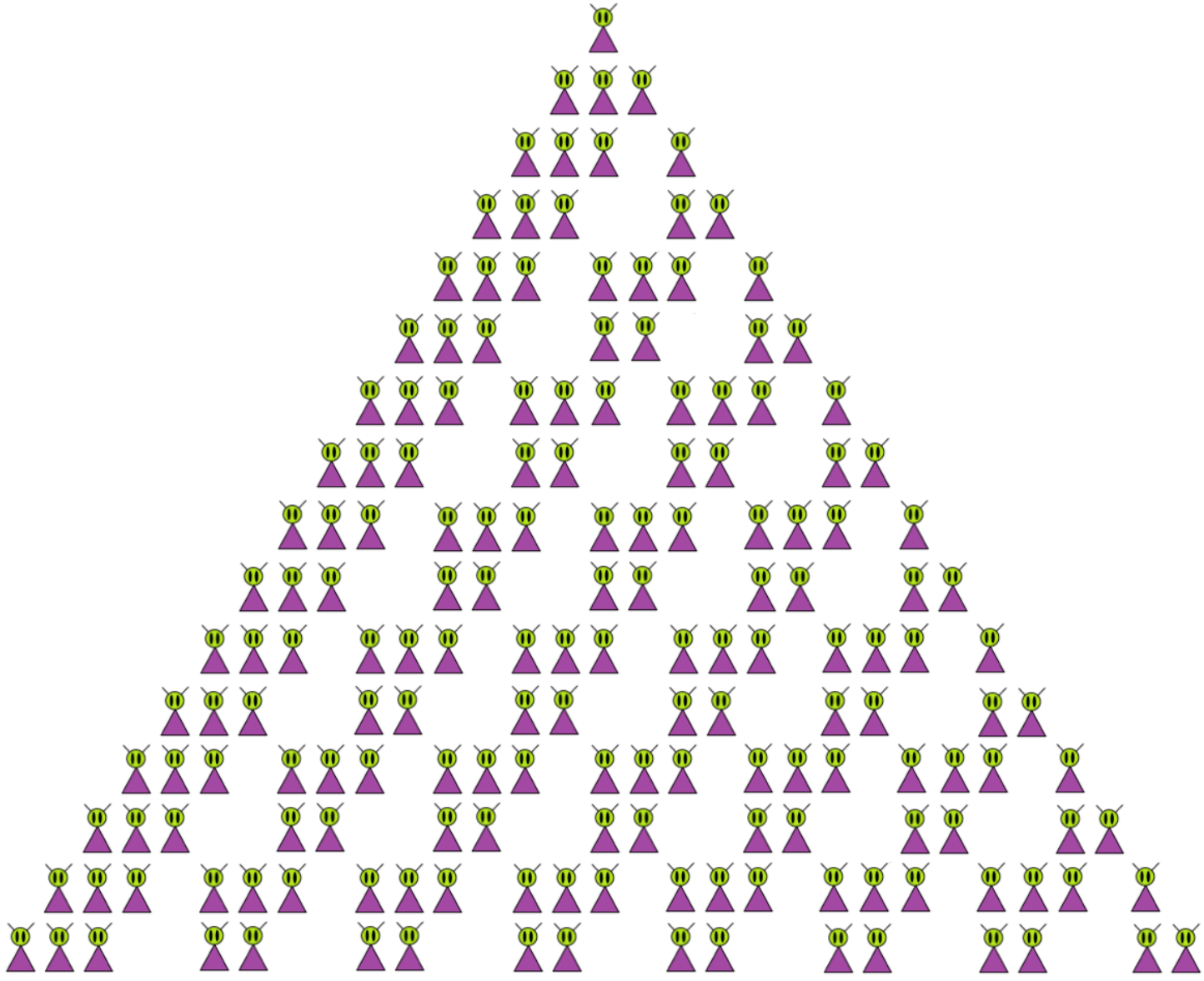
What about different generation rules?



What about different generation rules?



Exercise: Starting with a single Budling, use these new rules to find their state after ten days.
(Don't just calculate — find the pattern!)



Less aliens, more colors

Less aliens, more colors

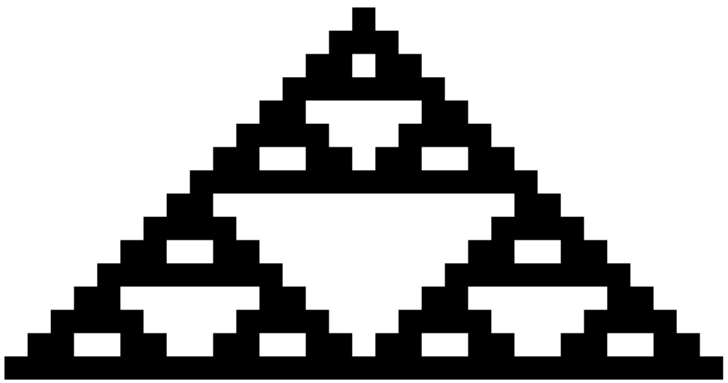
It's easier to see what's going on if we use colors.

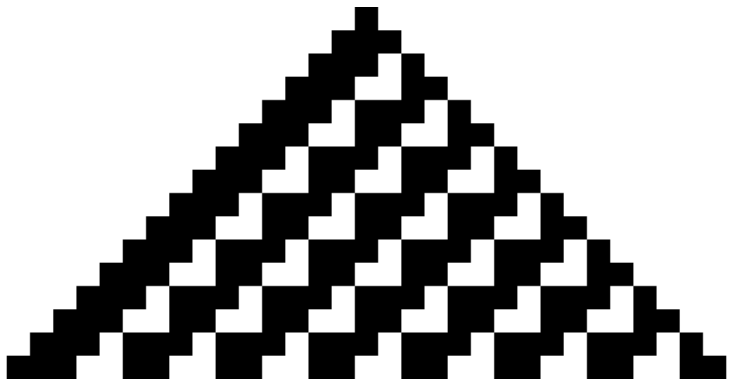
- Live alien $\rightarrow$ Black cell
- Dead alien $\rightarrow$ White cell

Less aliens, more colors

It's easier to see what's going on if we use colors.

- Live alien $\rightarrow$ Black cell
- Dead alien $\rightarrow$ White cell

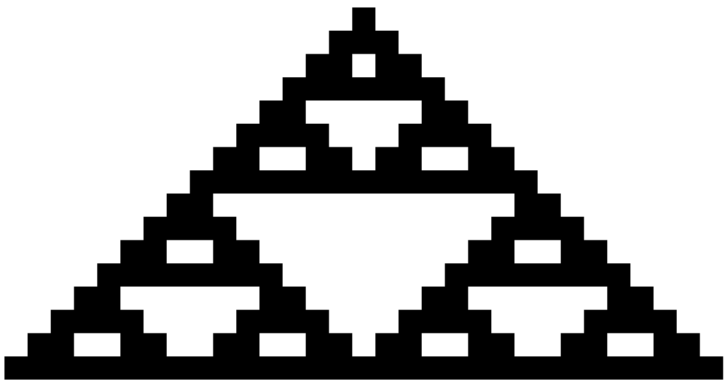


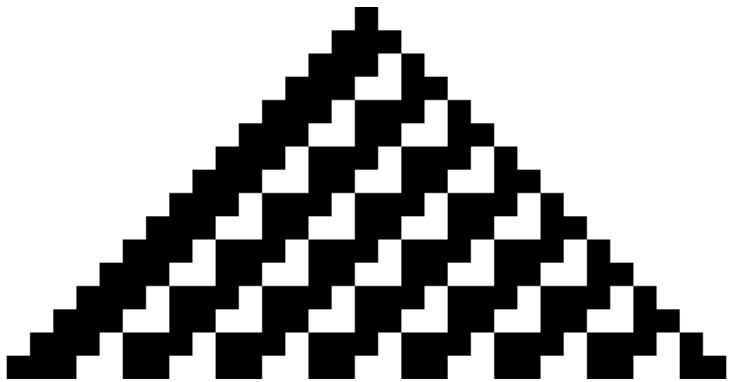


Less aliens, more colors

It's easier to see what's going on if we use colors.

- Live alien $\rightarrow$ Black cell
- Dead alien $\rightarrow$ White cell





These patterns of cells are called **cellular automata**.

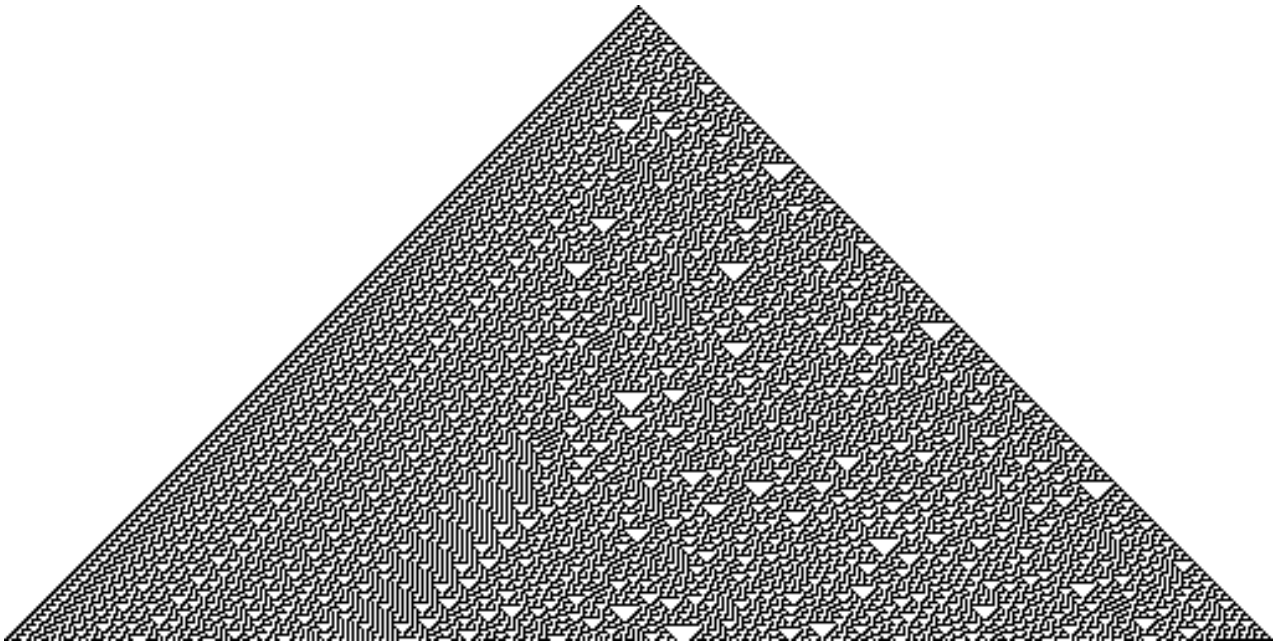
Cellular Automata

Cellular Automata

A **cellular automaton** starts at an initial state of live and dead cells, and uses *transition rules* to create a new state.

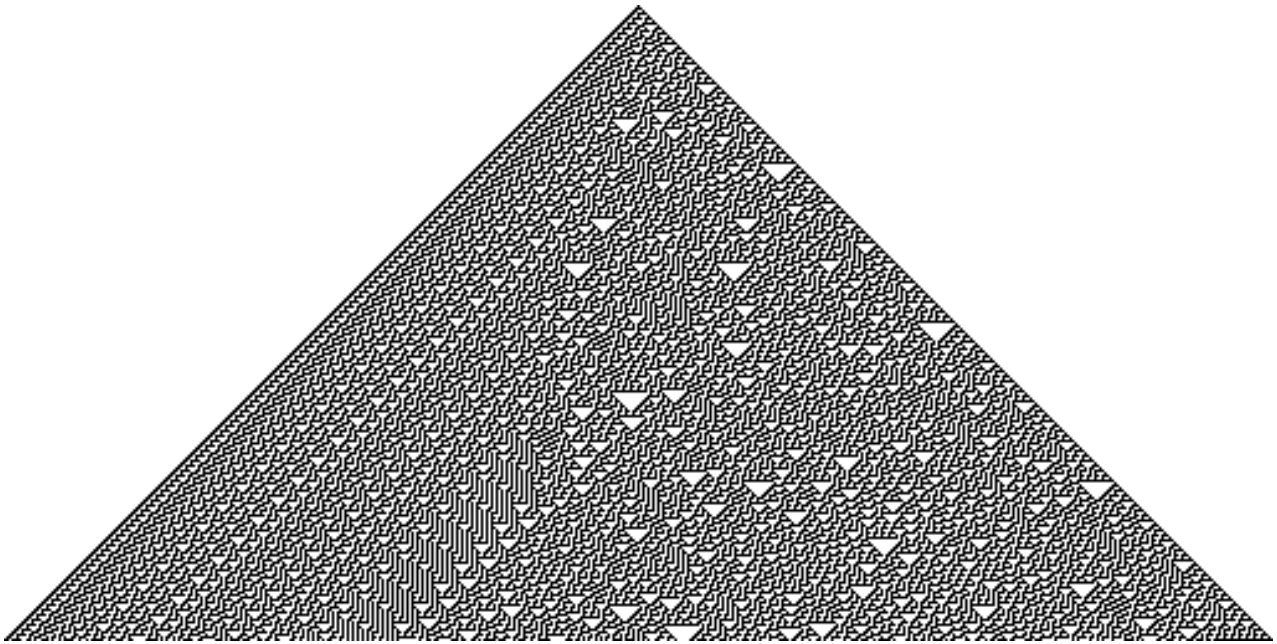
Cellular Automata

A **cellular automaton** starts at an initial state of live and dead cells, and uses *transition rules* to create a new state.



Cellular Automata

A **cellular automaton** starts at an initial state of live and dead cells, and uses *transition rules* to create a new state.



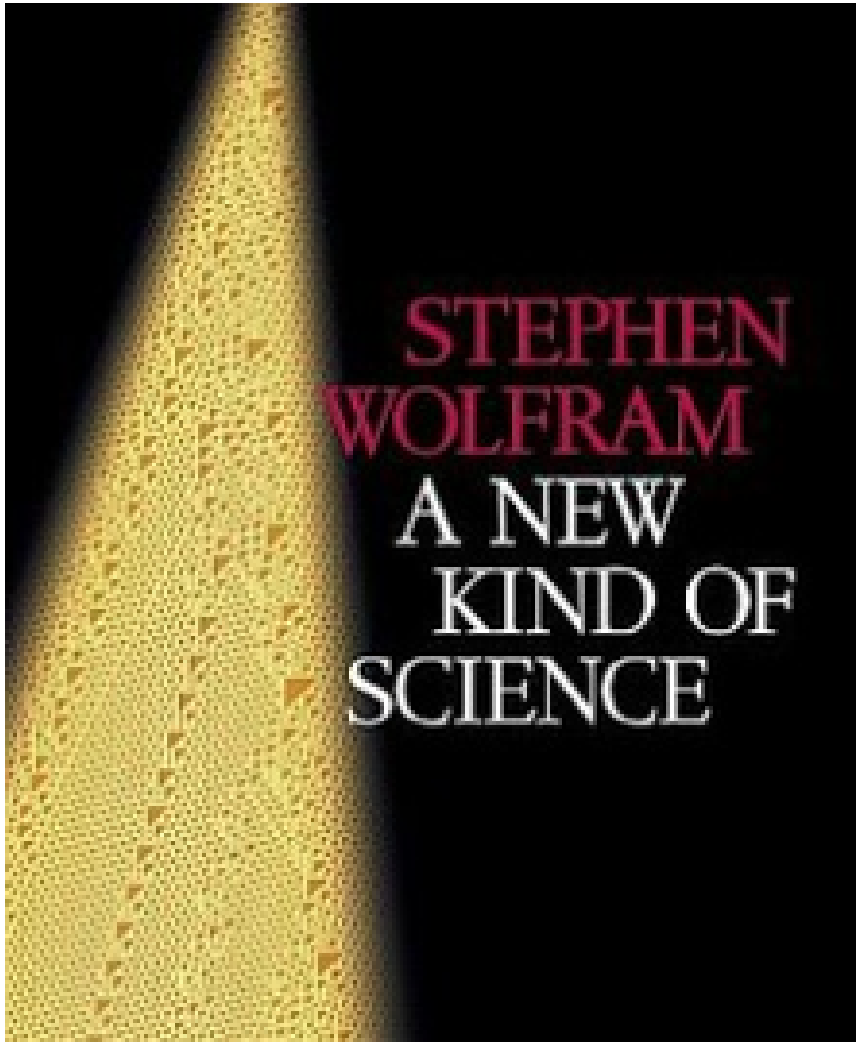
Even simple transition rules can generate chaotic patterns.

Stephen Wolfram

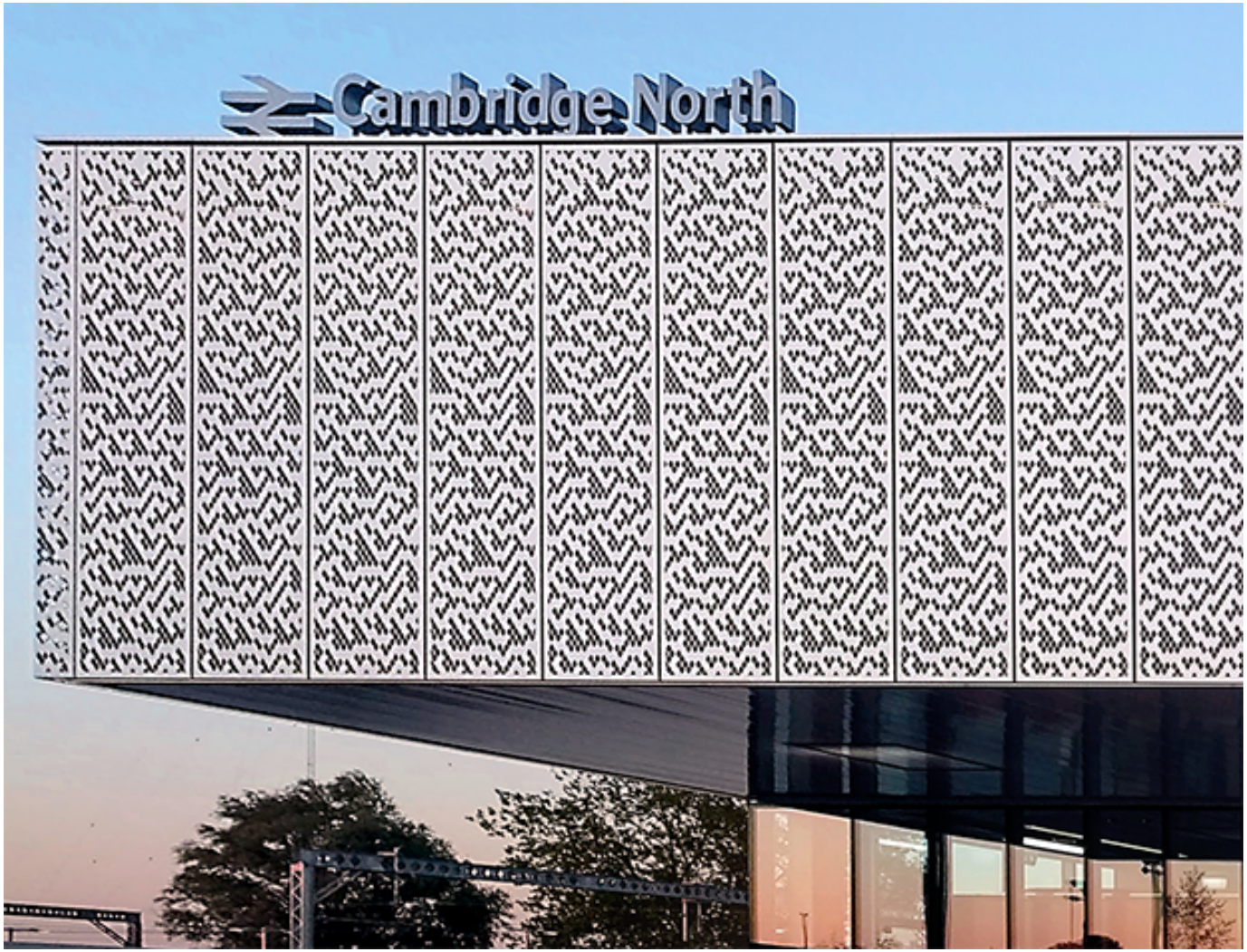
Creator of Mathematica and Elementary Cellular Automata
[VIDEO]

A New Kind Of Science

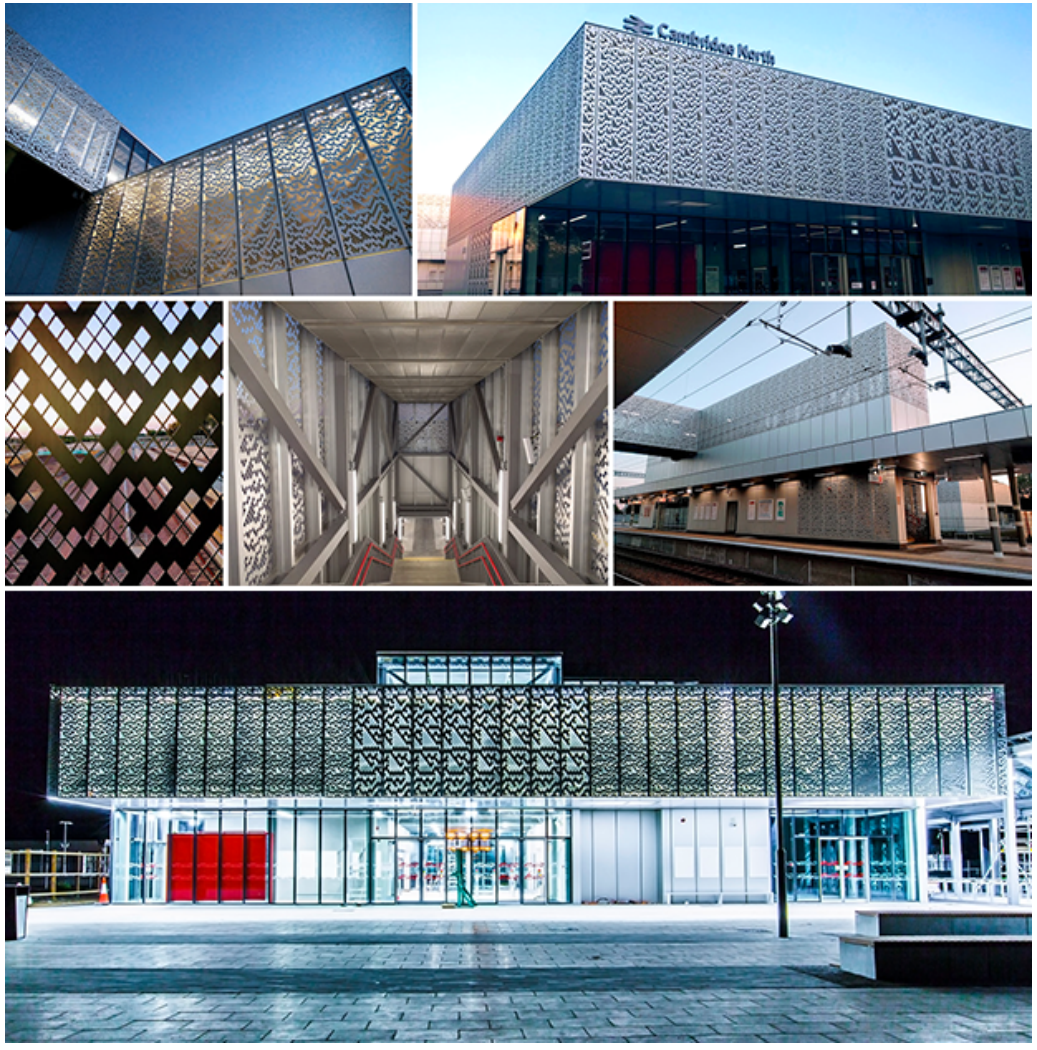
Free online: https://www.wolframscience.com/nks/



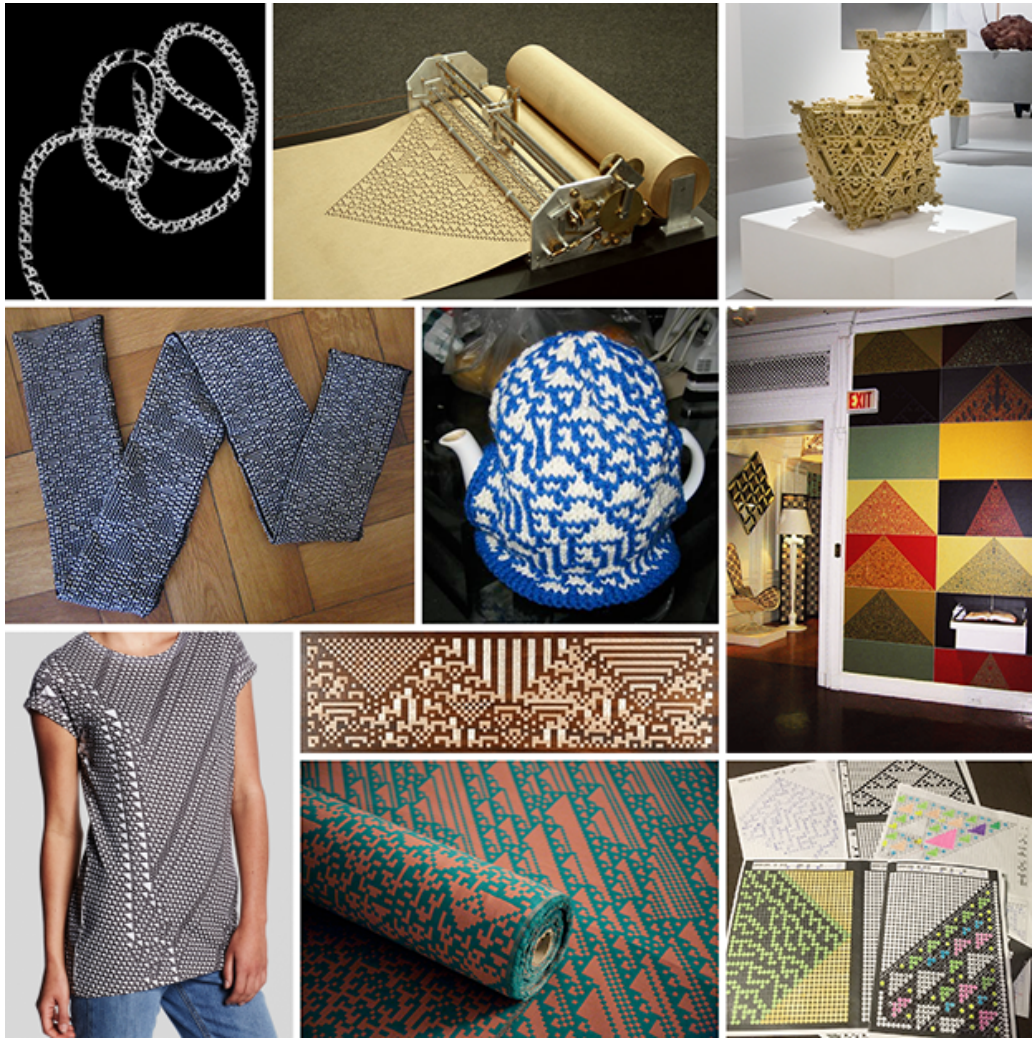
Automatic patterns in real life



Automatic patterns in real life

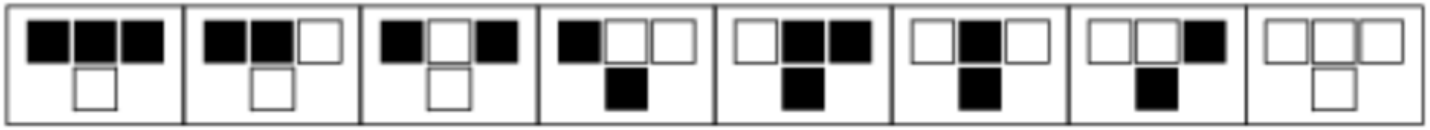


Automatic patterns in real life



Rule 30: A closer look

Rule 30: A closer look



Rule 30: A closer look

This is an example of an **elementary cellular automaton**: two states (black/white), and the state of each cell only depends on the cells to its left and right.

Rule 30: A closer look

This is an example of an **elementary cellular automaton**: two states (black/white), and the state of each cell only depends on the cells to its left and right.

Exercise: Iterate Rule 30 ten times, starting with just one black cell.

Fun With Programming

Fun With Programming

It's probably easier to get a computer to do this.

Fun With Programming

It's probably easier to get a computer to do this.

Go to **WolframAlpha** and use Wolfram Language to compute!

Fun With Programming

It's probably easier to get a computer to do this.

Go to **WolframAlpha** and use Wolfram Language to compute!

Enter this:

```
CellularAutomaton[30, {{1}, 0}, 40]
```

Try different rules and initial states!

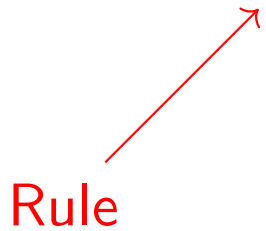
Fun With Programming

It's probably easier to get a computer to do this.

Go to **WolframAlpha** and use Wolfram Language to compute!

Enter this:

```
CellularAutomaton[30, {{1}, 0}, 40]
```



Try different rules and initial states!

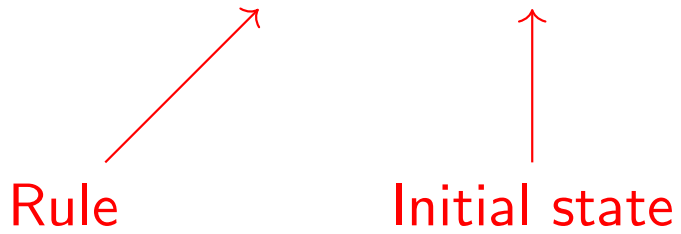
Fun With Programming

It's probably easier to get a computer to do this.

Go to **WolframAlpha** and use Wolfram Language to compute!

Enter this:

```
CellularAutomaton[30, {{1}, 0}, 40]
```



Try different rules and initial states!

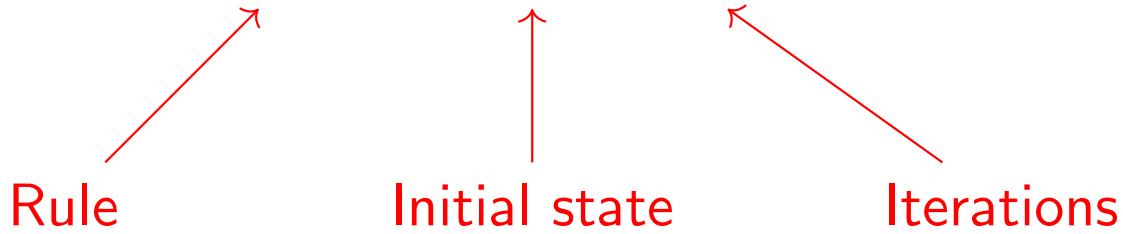
Fun With Programming

It's probably easier to get a computer to do this.

Go to **WolframAlpha** and use Wolfram Language to compute!

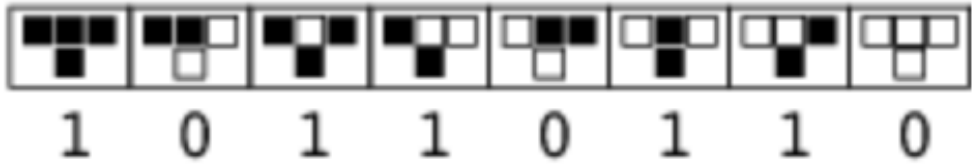
Enter this:

```
CellularAutomaton[30, {{1}, 0}, 40]
```

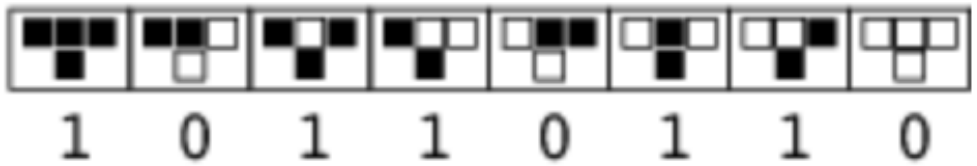


Try different rules and initial states!

Why is it called “Rule 30”?

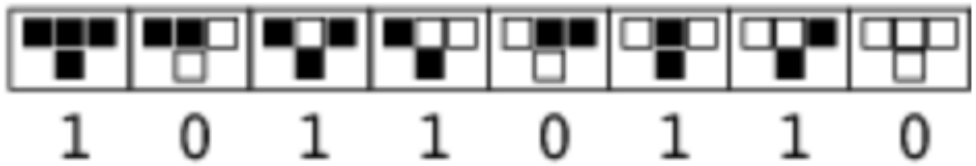


Why is it called “Rule 30”?



Wolfram calls this Rule 182 because 10110110 is the binary representation of the number 182.

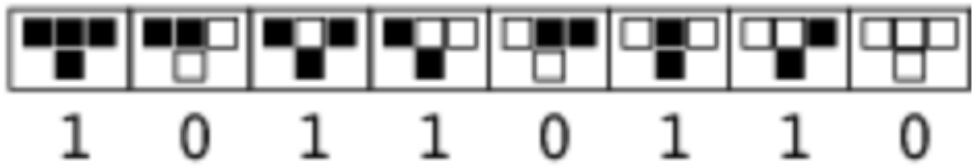
Why is it called "Rule 30"?



Wolfram calls this Rule 182 because 10110110 is the binary representation of the number 182.

$$(\mathbf{1})2^7+(\mathbf{0})2^6+(\mathbf{1})2^5+(\mathbf{1})2^4+(\mathbf{0})2^3+(\mathbf{1})2^2+(\mathbf{1})2^1+(\mathbf{0})2^0=182.$$

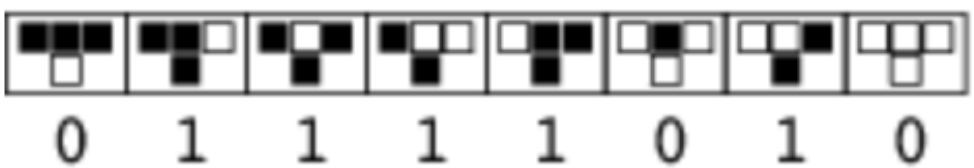
Why is it called "Rule 30"?



Wolfram calls this Rule 182 because 10110110 is the binary representation of the number 182.

$$(\mathbf{1})2^7+(\mathbf{0})2^6+(\mathbf{1})2^5+(\mathbf{1})2^4+(\mathbf{0})2^3+(\mathbf{1})2^2+(\mathbf{1})2^1+(\mathbf{0})2^0=182.$$

Another example: the binary expansion of 122 is 1111010, so Rule 122 is



Exercises

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.
- The following transition rules represent which rule number?

- How many elementary cellular automata are there in total?

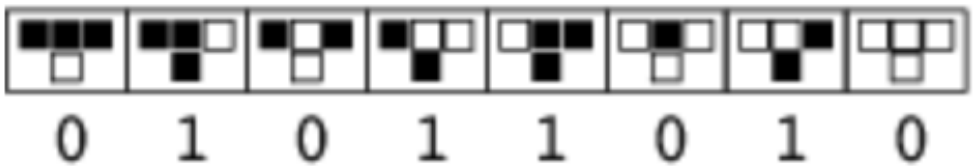
Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

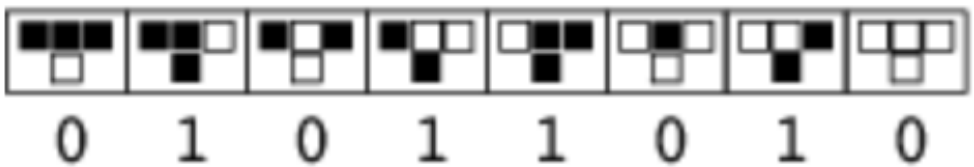
  Solution: $90 = 01011010$



Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

  Solution: $90 = 01011010$

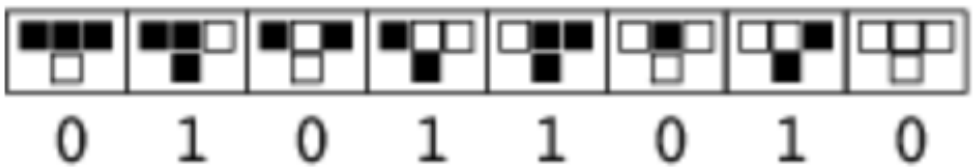


- Which cellular automaton has the following transition rules?

Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

  Solution: $90 = 01011010$



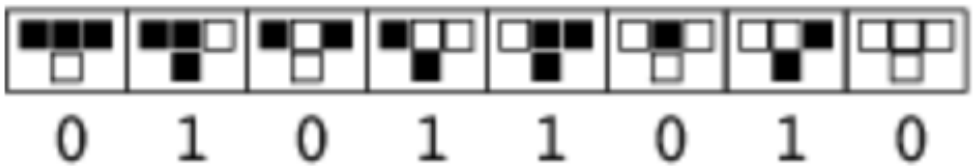
- Which cellular automaton has the following transition rules?

  Solution: $10111110 = 190$, so this is Rule 190.

Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

  Solution: 90 = 01011010



- Which cellular automaton has the following transition rules?

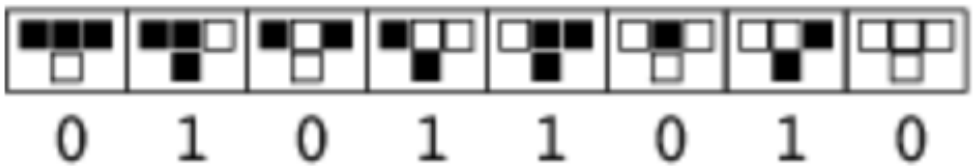
  Solution: 10111110 = 190, so this is Rule 190.

- How many elementary cellular automata are there in total?

Solutions

- Find the binary expansion of 90, and use it to write down the transition rules for Rule 90.

  Solution: $90 = 01011010$



- Which cellular automaton has the following transition rules?

  Solution: $10111110 = 190$, so this is Rule 190.

- How many elementary cellular automata are there in total?

  Solution: There are $2^8 = 256$!

Changing the initial state

Changing the initial state

So far we have started with just a single live cell.

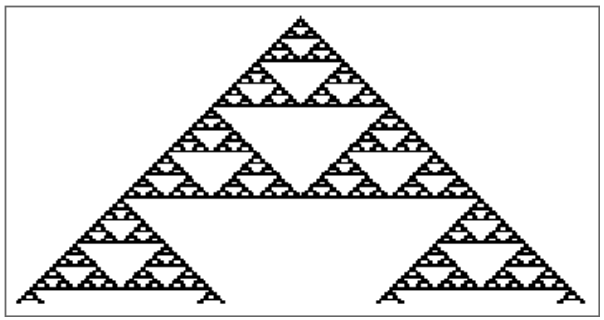
Changing the initial state

So far we have started with just a single live cell. What if we start with several?

Changing the initial state

So far we have started with just a single live cell. What if we start with several?

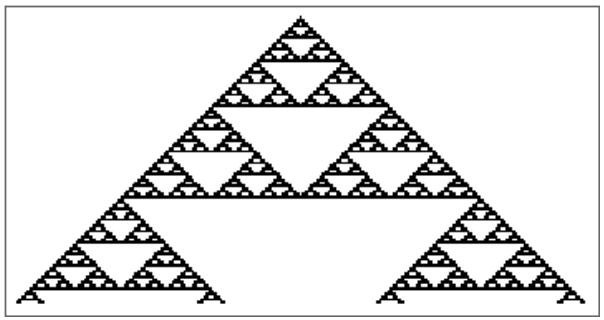
One live state



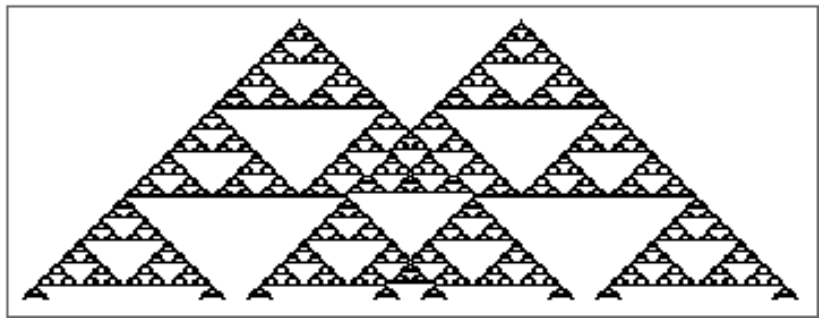
Changing the initial state

So far we have started with just a single live cell. What if we start with several?

One live state



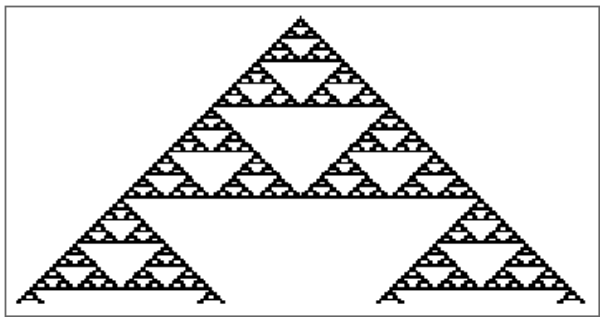
Two live states



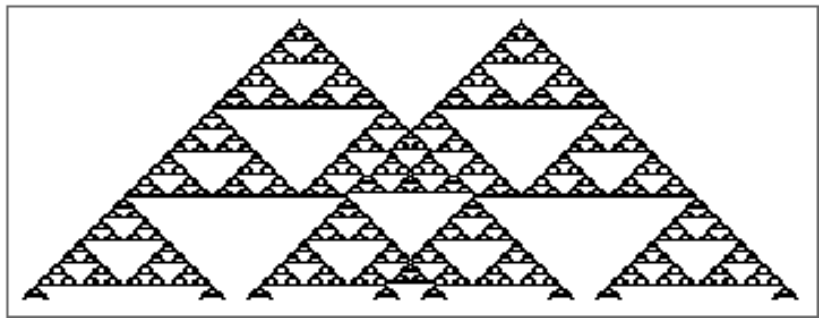
Changing the initial state

So far we have started with just a single live cell. What if we start with several?

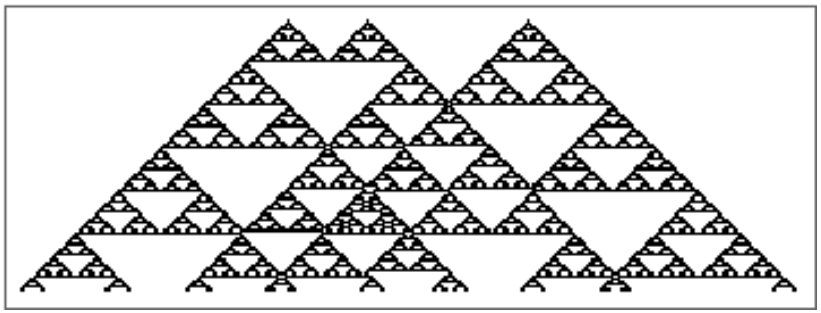
One live state



Two live states



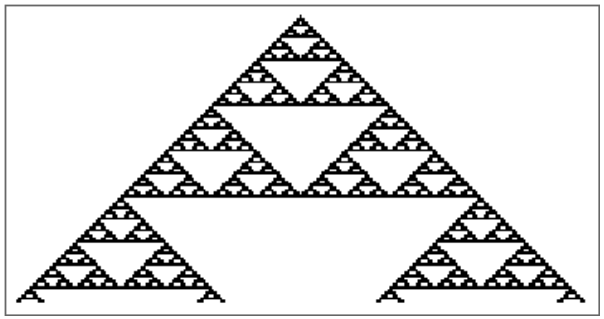
Three live states



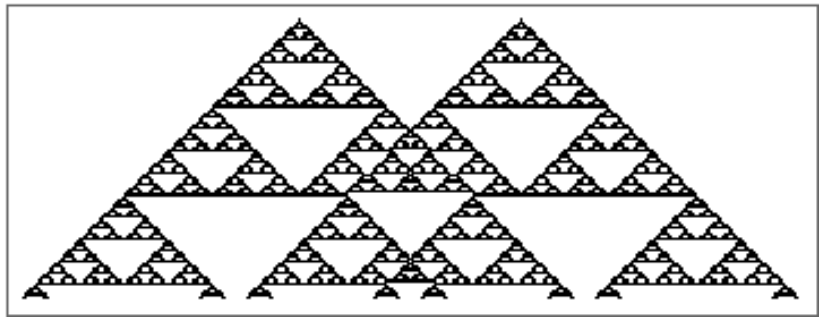
Changing the initial state

So far we have started with just a single live cell. What if we start with several?

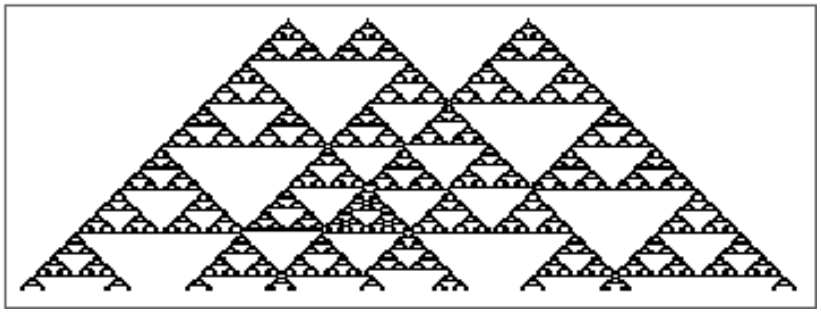
One live state



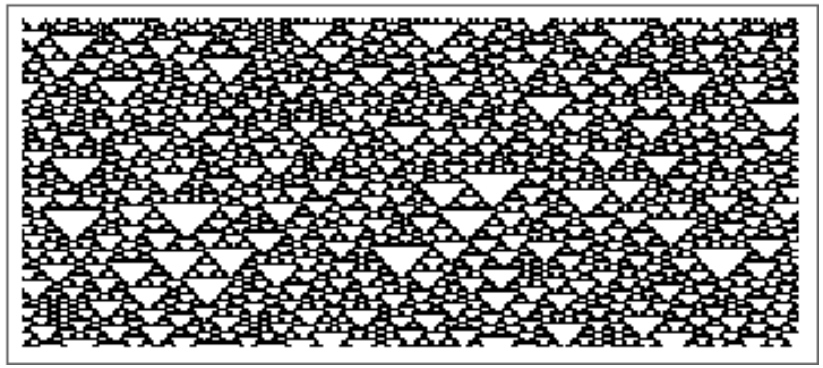
Two live states



Three live states



Random



Changing the initial state

Changing the initial state

- Single live cell:

Changing the initial state

- Single live cell:

```
CellularAutomaton[173, {{1},0}, 150]
```

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

  CellularAutomaton[173, {{1,0,1,0,0,1},0}, 150]

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

  CellularAutomaton[173, {{1,0,1,0,0,1},0}, 150]

  $\Longrightarrow$ Initial state is $\cdots 000101001000 \cdots$

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

  CellularAutomaton[173, {{1,0,1,0,0,1},0}, 150]

  $\Longrightarrow$ Initial state is $\cdots 000101001000 \cdots$

- Random state:

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

  CellularAutomaton[173, {{1,0,1,0,0,1},0}, 150]

  $\Longrightarrow$ Initial state is $\cdots 000101001000 \cdots$

- Random state:

  RandomInteger[1, 250]

Changing the initial state

- Single live cell:

  CellularAutomaton[173, {{1},0}, 150]

  $\Longrightarrow$ Runs Rule 173 with 150 iterations, starting with initial state $\cdots 0001000 \cdots$

- Block padded with zeros:

  CellularAutomaton[173, {{1,0,1,0,0,1},0}, 150]

  $\Longrightarrow$ Initial state is $\cdots 000101001000 \cdots$

- Random state:

  RandomInteger[1, 250]

  $\Longrightarrow$ Randomly generates a sequence of 250 zeros and ones.

Why Mathematicians Love Cellular Automata

So far, we have seen how to *generate* cellular automata using transition rules.

Why Mathematicians Love Cellular Automata

So far, we have seen how to *generate* cellular automata using transition rules.

But aside from making cool shower curtains, what's interesting about these?

Why Mathematicians Love Cellular Automata

So far, we have seen how to *generate* cellular automata using transition rules.

But aside from making cool shower curtains, what's interesting about these?

One of the most interesting subjects in cellular automata — with many unsolved problems today! — is **Gardens Of Eden**.

A Garden Of Eden in Rule 126

Iterate Rule 126 starting with a single live cell.

A Garden Of Eden in Rule 126

Iterate Rule 126 starting with a single live cell.

Question: If you keep iterating Rule 126, will you ever reach a single live cell again?

A Garden Of Eden in Rule 126

Iterate Rule 126 starting with a single live cell.

Question: If you keep iterating Rule 126, will you ever reach a single live cell again?

What if you start with two live cells — $\cdots 000101000 \cdots$

A Garden Of Eden in Rule 126

Iterate Rule 126 starting with a single live cell.

Question: If you keep iterating Rule 126, will you ever reach a single live cell again?

What if you start with two live cells — $\cdots 000101000 \cdots$

Or three? $\cdots 00010101000 \cdots$

A Garden Of Eden in Rule 126

Iterate Rule 126 starting with a single live cell.

Question: If you keep iterating Rule 126, will you ever reach a single live cell again?

What if you start with two live cells — $\cdots 000101000 \cdots$

Or three? $\cdots 00010101000 \cdots$

Is there *any* initial state that will ever result in a single live state?

Rule 126

In fact, there is NO initial state that will *ever* result in the state $\cdots 0001000 \cdots$ (using Rule 126).

Rule 126

In fact, there is NO initial state that will *ever* result in the state $\cdots 0001000 \cdots$ (using Rule 126).

Why not? We can *prove* it mathematically!

Garden Of Eden

A **predecessor** of a state X is another state that will eventually lead to X.

Garden Of Eden

A **predecessor** of a state X is another state that will eventually lead to X.

A **Garden Of Eden** is a state with no predecessor.

Garden Of Eden

A **predecessor** of a state X is another state that will eventually lead to X.

A **Garden Of Eden** is a state with no predecessor.

Garden Of Eden

A **predecessor** of a state X is another state that will eventually lead to X.

A **Garden Of Eden** is a state with no predecessor.

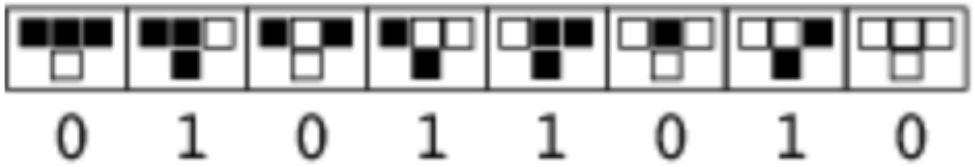
What we just saw is that Rule 126 has a Garden Of Eden: a single live state!

Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.

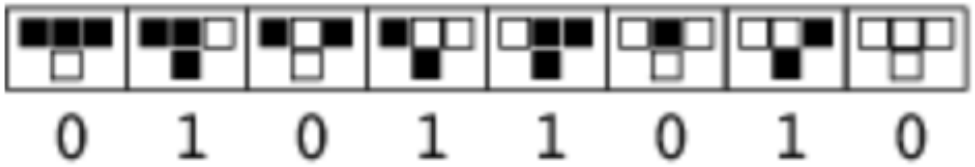
Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.



Garden Of Eden in Rule 90?

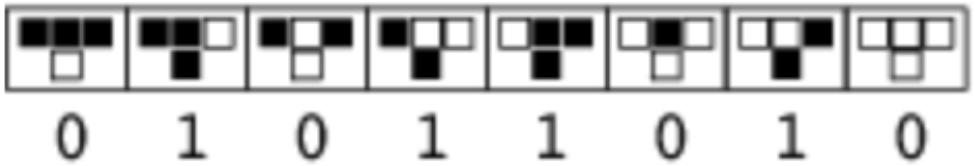
Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.



Rule 90 is known to have *no* Garden Of Eden: this means every state has a predecessor.

Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.

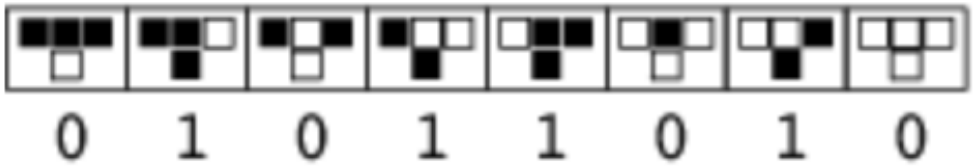


Rule 90 is known to have *no* Garden Of Eden: this means every state has a predecessor.

Exercise: Find a predecessor for each of the following states.

Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.



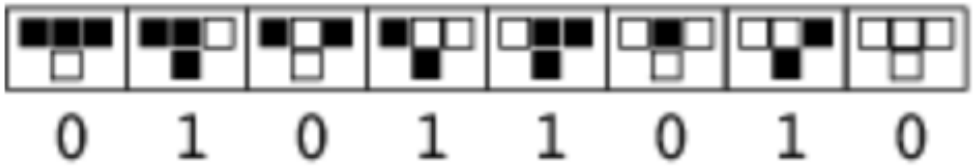
Rule 90 is known to have *no* Garden Of Eden: this means every state has a predecessor.

Exercise: Find a predecessor for each of the following states.

(a) 00011000

Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.



Rule 90 is known to have *no* Garden Of Eden: this means every state has a predecessor.

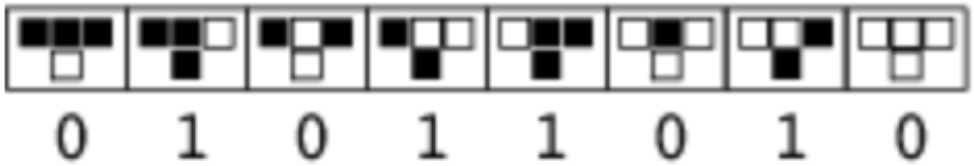
Exercise: Find a predecessor for each of the following states.

(a) 00011000

(b) 000111000

Garden Of Eden in Rule 90?

Rule 90 is called the **XOR** automaton, because a cell's state at time $t+1$ is the **exclusive or** of its neighboring states at time t.



Rule 90 is known to have *no* Garden Of Eden: this means every state has a predecessor.

Exercise: Find a predecessor for each of the following states.

(a) 00011000

(b) 000111000

(c) 000101100111000

Twin states

Twin states

A cellular automaton has **twin states** if it has two states that eventually lead to the same outcome.

Twin states

A cellular automaton has **twin states** if it has two states that eventually lead to the same outcome.

Exercise: Iterate Rule 90 (the XOR automaton) five times on each of these initial states.

Twin states

A cellular automaton has **twin states** if it has two states that eventually lead to the same outcome.

Exercise: Iterate Rule 90 (the XOR automaton) five times on each of these initial states.

(a) $\cdots 0001000 \cdots$, *i.e.* 

Twin states

A cellular automaton has **twin states** if it has two states that eventually lead to the same outcome.

Exercise: Iterate Rule 90 (the XOR automaton) five times on each of these initial states.

(a) $\cdots 0001000 \cdots$, *i.e.* 

(b) $\cdots 000101000 \cdots$. *i.e.* 

Twin states

A cellular automaton has **twin states** if it has two states that eventually lead to the same outcome.

Exercise: Iterate Rule 90 (the XOR automaton) five times on each of these initial states.

(a) $\cdots 0001000 \cdots$, *i.e.* 

(b) $\cdots 000101000 \cdots$. *i.e.* 

What did you notice?

Twin states in Rule 90

In Rule 90, 00100 and 01010 are twin states!

Twin states in Rule 90

In Rule 90, 00100 and 01010 are twin states!

Here is another pair of twin states for Rule 90: 10101 and 101000101.

The Garden Of Eden Theorem

Incredible fact about cellular automaton:

The Garden Of Eden Theorem

Incredible fact about cellular automaton: any CA with twin states must also have a Garden Of Eden.

The Garden Of Eden Theorem

Incredible fact about cellular automaton: any CA with twin states must also have a Garden Of Eden.

Conversely, any CA with a Garden Of Eden must also have twin states!

The Garden Of Eden Theorem

Incredible fact about cellular automaton: any CA with twin states must also have a Garden Of Eden.

Conversely, any CA with a Garden Of Eden must also have twin states!

Theorem (Moore–Myhill, 1960's). *A cellular automaton has a Garden Of Eden if and only if it has twin states.*

The Garden Of Eden Theorem

Incredible fact about cellular automaton: any CA with twin states must also have a Garden Of Eden.

Conversely, any CA with a Garden Of Eden must also have twin states!

Theorem (Moore–Myhill, 1960's). *A cellular automaton has a Garden Of Eden if and only if it has twin states.*

For example, we saw a Garden Of Eden *and* twin states for Rule 90.

The Garden Of Eden Theorem

Incredible fact about cellular automaton: any CA with twin states must also have a Garden Of Eden.

Conversely, any CA with a Garden Of Eden must also have twin states!

Theorem (Moore–Myhill, 1960's). *A cellular automaton has a Garden Of Eden if and only if it has twin states.*

For example, we saw a Garden Of Eden *and* twin states for Rule 90.

On the other hand, Rule 126 has a Garden Of Eden, so it **must** have twin states!

Multistate automata

Multistate automata

So far, we only allowed cells to be ALIVE or DEAD (1 or 0).

Multistate automata

So far, we only allowed cells to be ALIVE or DEAD (1 or 0). These are called the **elementary cellular automata**, and there are 256 of them.

Multistate automata

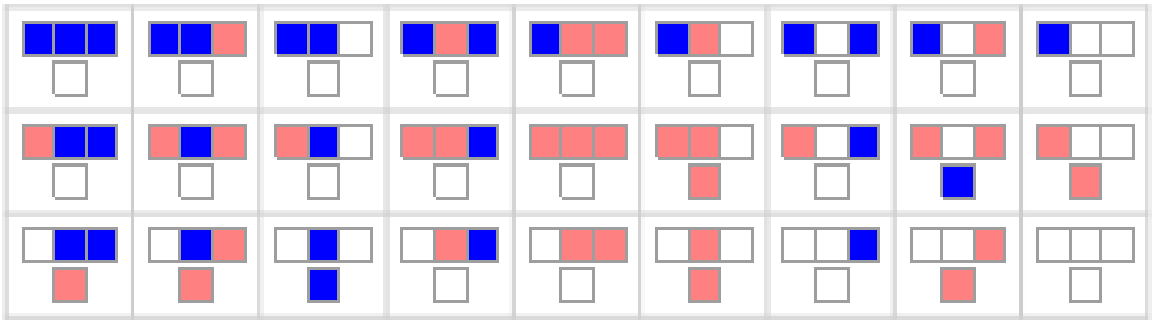
So far, we only allowed cells to be ALIVE or DEAD (1 or 0). These are called the **elementary cellular automata**, and there are 256 of them.

What about three states?

Multistate automata

So far, we only allowed cells to be ALIVE or DEAD (1 or 0). These are called the **elementary cellular automata**, and there are 256 of them.

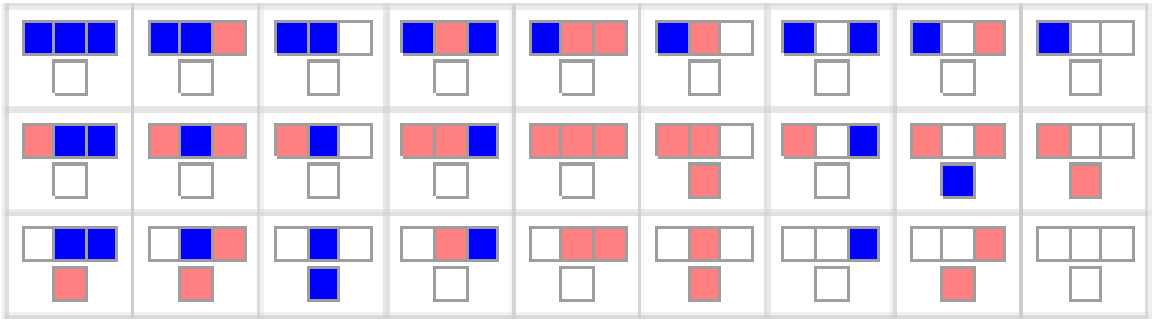
What about three states?

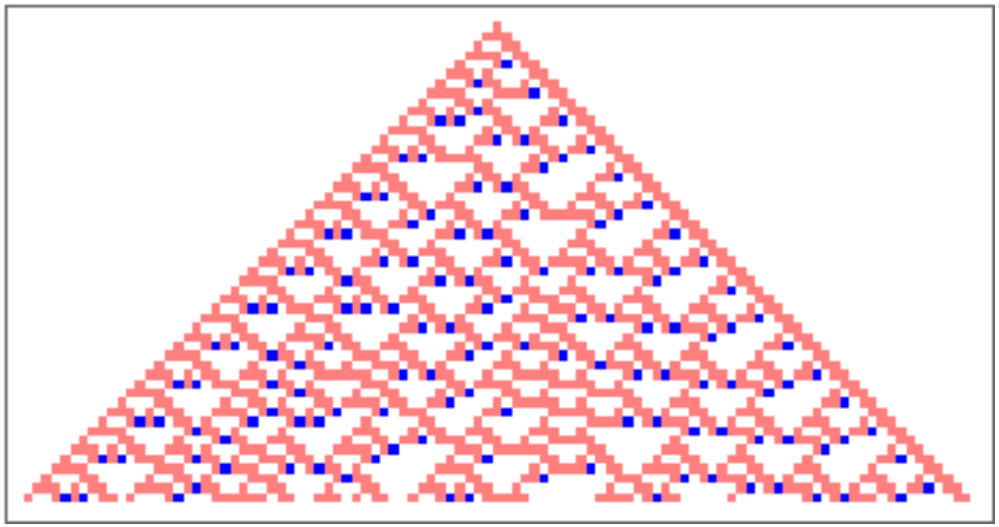


Multistate automata

So far, we only allowed cells to be ALIVE or DEAD (1 or 0). These are called the **elementary cellular automata**, and there are 256 of them.

What about three states?





Multistate automata

Some interesting 3-color automata:

- Rule 679458
- Rule 3333333
- Rule 21111

Multistate automata

Some interesting 3-color automata:

- Rule 679458
- Rule 3333333
- Rule 21111

How many 3-color rules are there?

Multistate automata

Some interesting 3-color automata:

- Rule 679458
- Rule 3333333
- Rule 21111

How many 3-color rules are there?

How many k-color rules are there?

Totalistic automata

Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

Totalistic automata

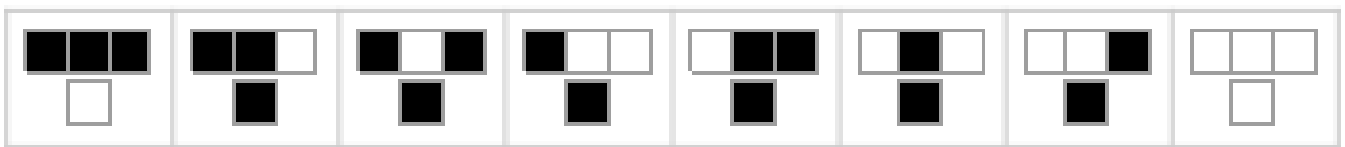
Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

Total	3	2	1	0
State	0	1	1	0

Totalistic automata

Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

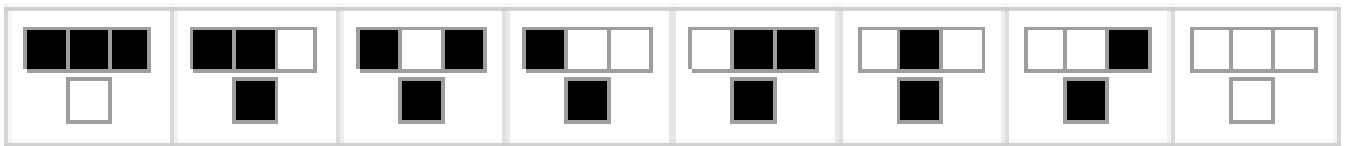
Total	3	2	1	0
State	0	1	1	0



Totalistic automata

Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

Total	3	2	1	0
State	0	1	1	0

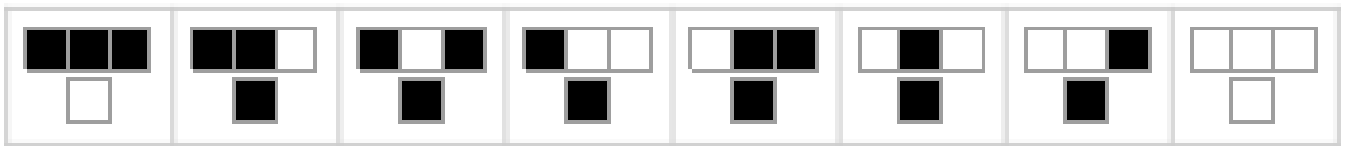


It's Rule 126! The friendship rule!

Totalistic automata

Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

Total	3	2	1	0
State	0	1	1	0



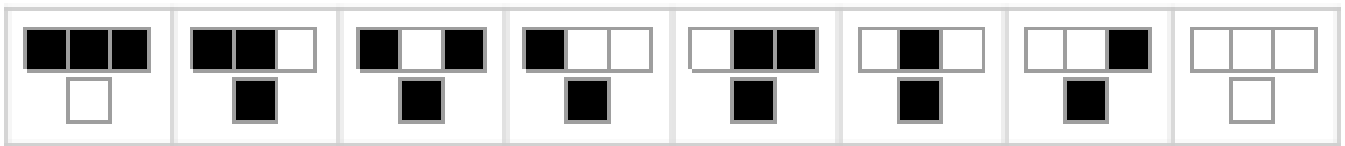
It's Rule 126! The friendship rule!

```
CellularAutomaton[{6, {2, 1}}]
```

Totalistic automata

Totalistic rules: the state of a cell only depends on the *total* of its neighboring cells.

Total	3	2	1	0
State	0	1	1	0



It's Rule 126! The friendship rule!

```
CellularAutomaton[{6, {2, 1}}]
```

How many 2-color totalistic rules are there?

Totalistic automata

What about for 3-color automata?

Totalistic automata

What about for 3-color automata?

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

Totalistic automata

What about for 3-color automata?

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

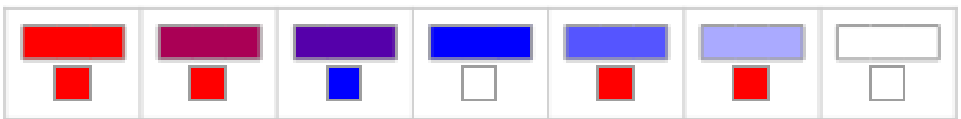
Total	6	5	4	3	2	1	0
State	2	2	1	0	2	2	0

Totalistic automata

What about for 3-color automata?

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

Total	6	5	4	3	2	1	0
State	2	2	1	0	2	2	0

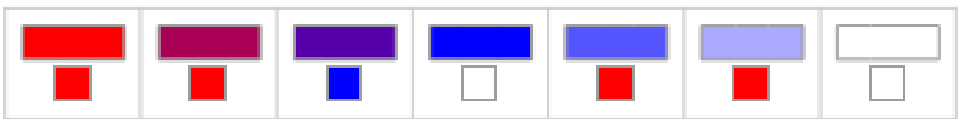


Totalistic automata

What about for 3-color automata?

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

Total	6	5	4	3	2	1	0
State	2	2	1	0	2	2	0



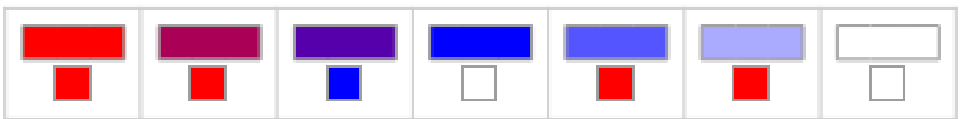
Notice 2049 is just 2210220 in base-3.

Totalistic automata

What about for 3-color automata?

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

Total	6	5	4	3	2	1	0
State	2	2	1	0	2	2	0



Notice 2049 is just 2210220 in base-3.

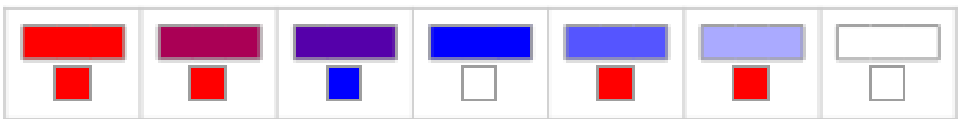
How many 3-color totalistic rules are there?

Totalistic automata

What about for 3-color automata?

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

Total	6	5	4	3	2	1	0
State	2	2	1	0	2	2	0



Notice 2049 is just 2210220 in base-3.

How many 3-color totalistic rules are there?

How many k-color totalistic rules are there?

Totalistic automata: A Code Challenge

Exercise: Using Wolfram Language, write code that generates a 4-color totalistic rule with initial state 102321.

See if you can find one that looks cool! Then use RulePlot to print out its transition rules.

Next week ...

So far, we have only studied automata where each state is represented by a line (1-dimensional).

Next week ...

So far, we have only studied automata where each state is represented by a line (1-dimensional).

What about 2-dimensional automata?

Next week ...

So far, we have only studied automata where each state is represented by a line (1-dimensional).

What about 2-dimensional automata?

Next week ...

So far, we have only studied automata where each state is represented by a line (1-dimensional).

What about 2-dimensional automata?

See you next time!