Cellular Automata: introduction

A cellular automaton (CA) is a **discrete dynamical system** that consists of an **infinite grid** of finite state nodes (**cells**) that change their states depending on the states of their neighbors, according to a **local update rule**.

The system is **synchronous** and **uniform**: All cells change their state simultaneously, using the same update rule.

The operation is repeated at **discrete time steps**.

Cellular automata are used, for example,

- **in physics** as discrete models of physical systems,
- **in computer science** as models of massively parallel computation under the realistic constraints of locality and uniformity,
- **in mathematics** as endomorphisms of the full shift in the context of symbolic dynamics.

Cellular automata possess several fundamental properties of the physical world: they are

- **massively parallel**,
- **homogeneous** (=uniform) in time and space,
- all interactions are **local**,
- time **reversibility** and **conservation laws** can be obtained by choosing the local update rule properly.

Famous example: Game-Of-Life

GOL was invented by John Conway in 1970.

- Infinite checker-board whose squares (=cells) are colored black (=**alive**) or white (=**dead**).
- At each discrete time step each cell counts the number of living cells surrounding it, and based on this number determines its new state.
- All cells change their state simultaneously.

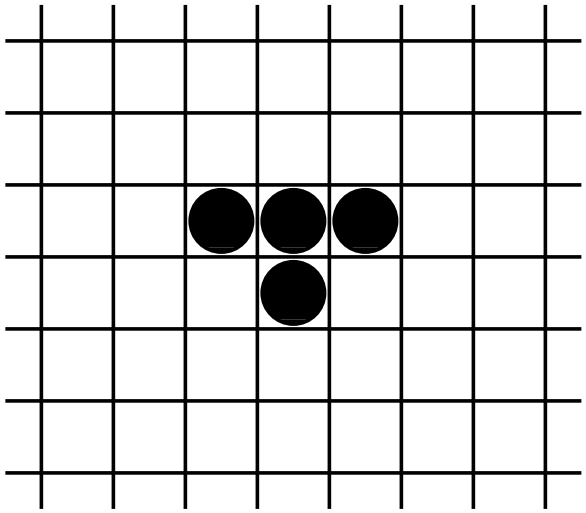
Each cell counts the living cells in the eight surrounding cells:

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.

All cells apply this simple local update rule simultaneously. As the process is repeated over and over again, a dynamical system is obtained that exhibits surprisingly complex behavior.

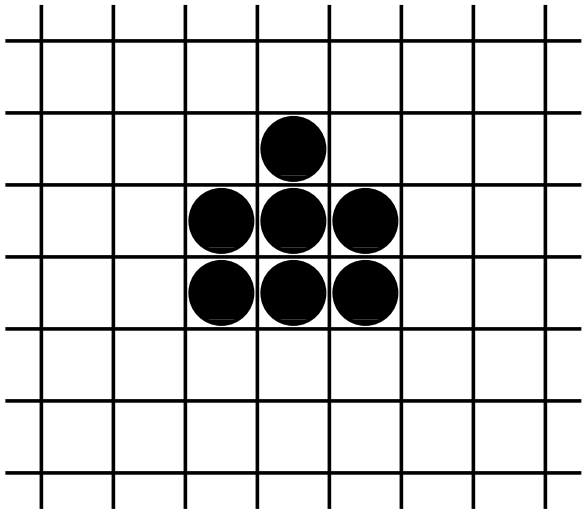
Each cell counts the living cells in the eight surrounding cells:

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



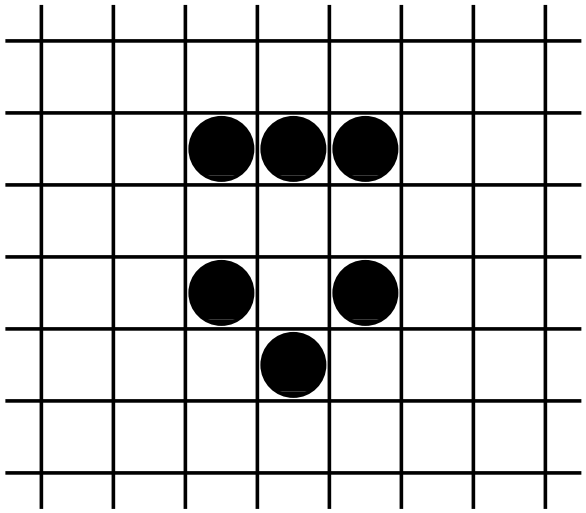
Each cell counts the living cells in the eight surrounding cells:

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



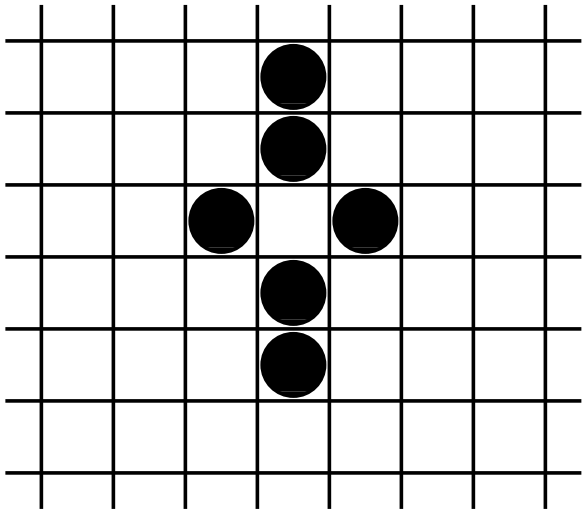
Each cell counts the living cells in the eight surrounding cells:

- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.

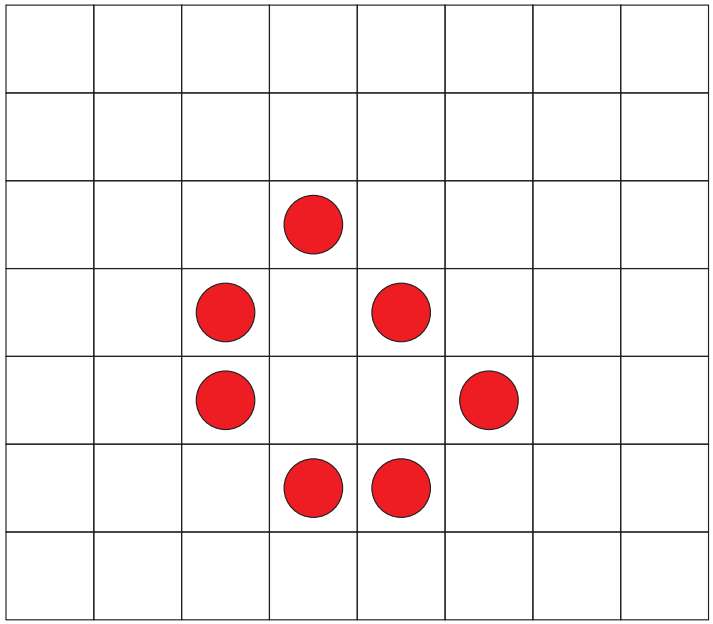


Each cell counts the living cells in the eight surrounding cells:

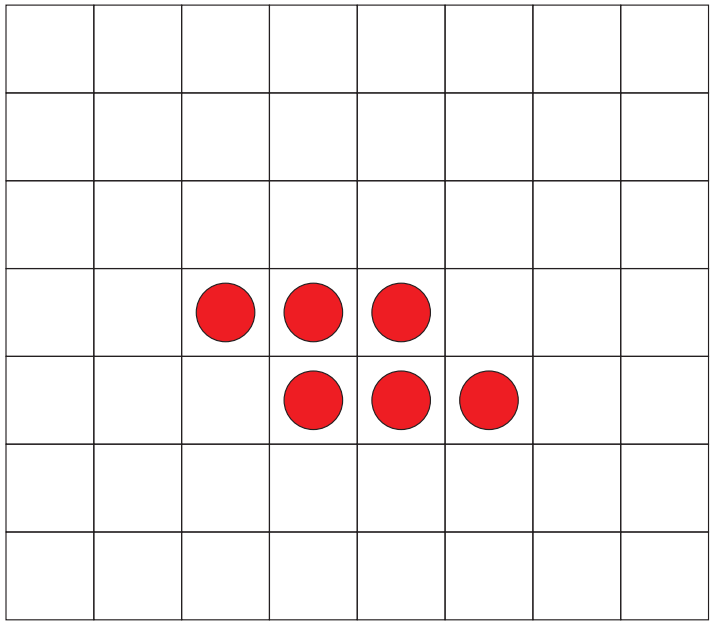
- If the cell is **alive** then it stays alive (survives) iff it has two or three live neighbors. Otherwise it dies of loneliness or overcrowding.
- If the cell is **dead** then it becomes alive iff it has exactly three living neighbors.



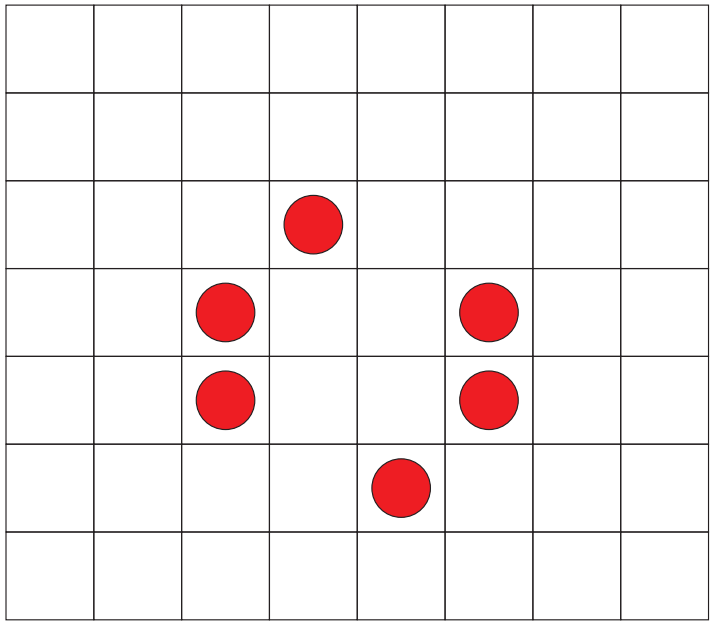
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**)



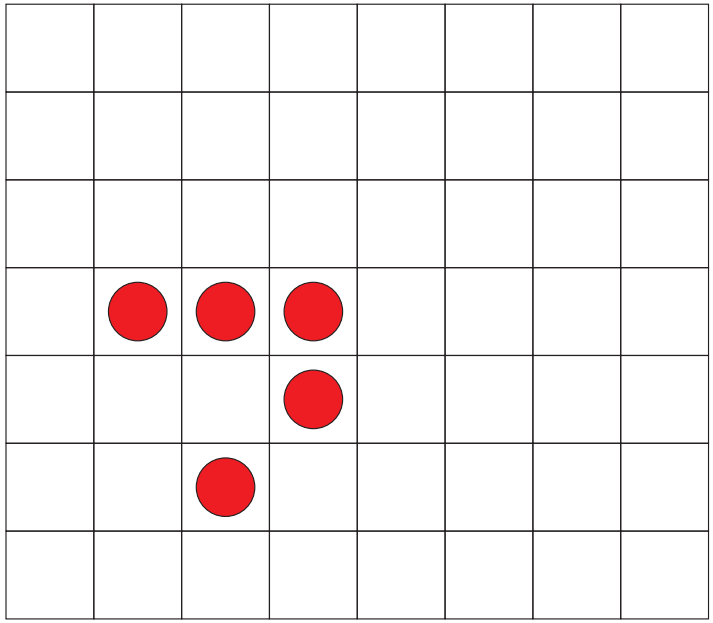
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**)



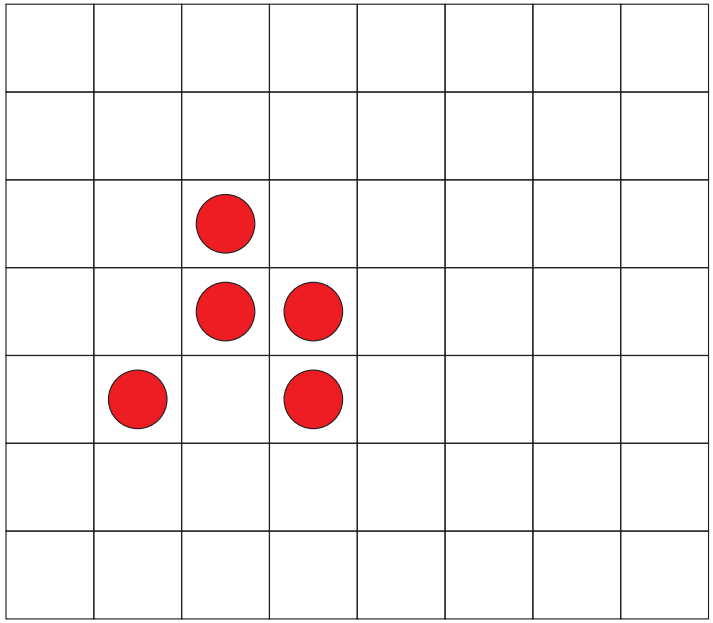
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**)



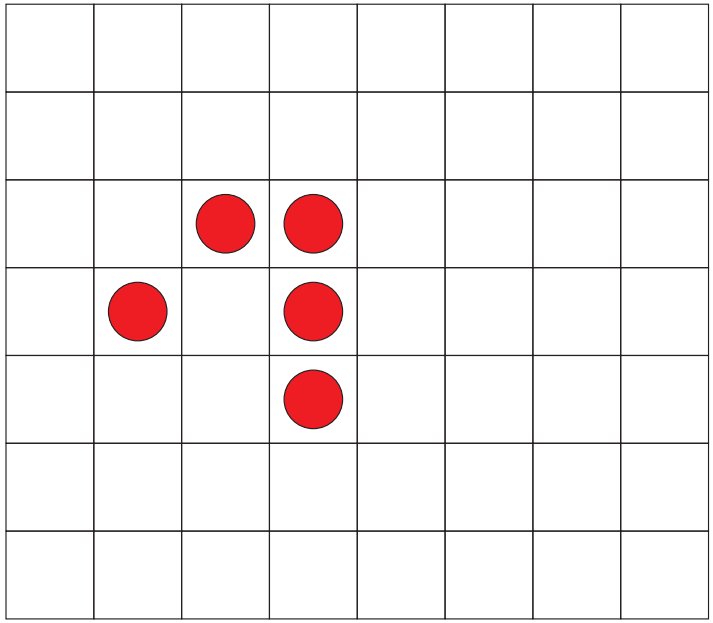
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



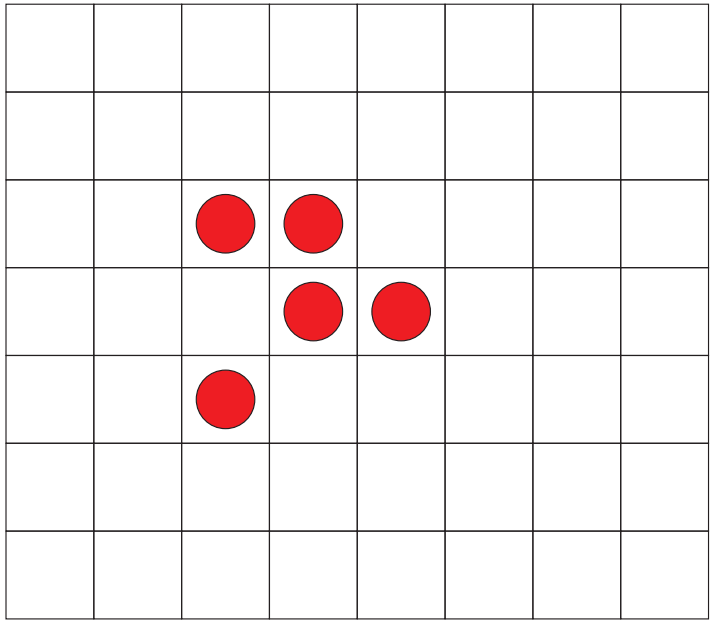
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



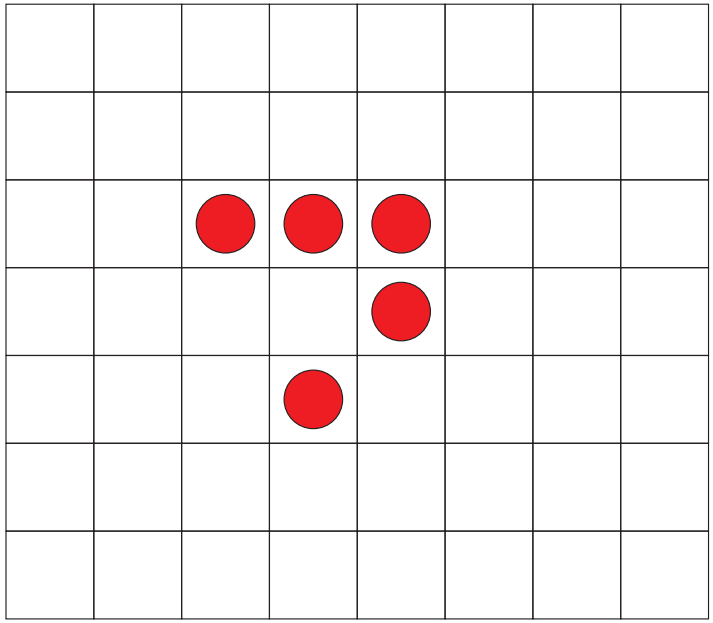
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



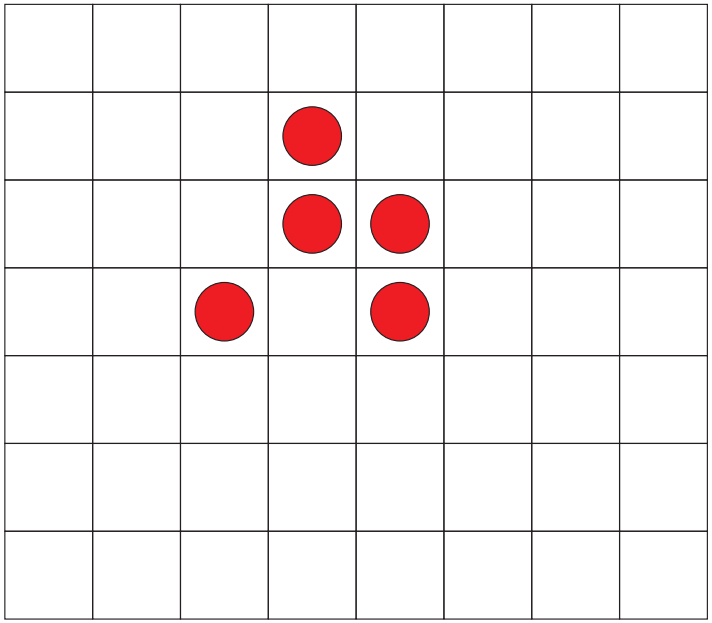
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



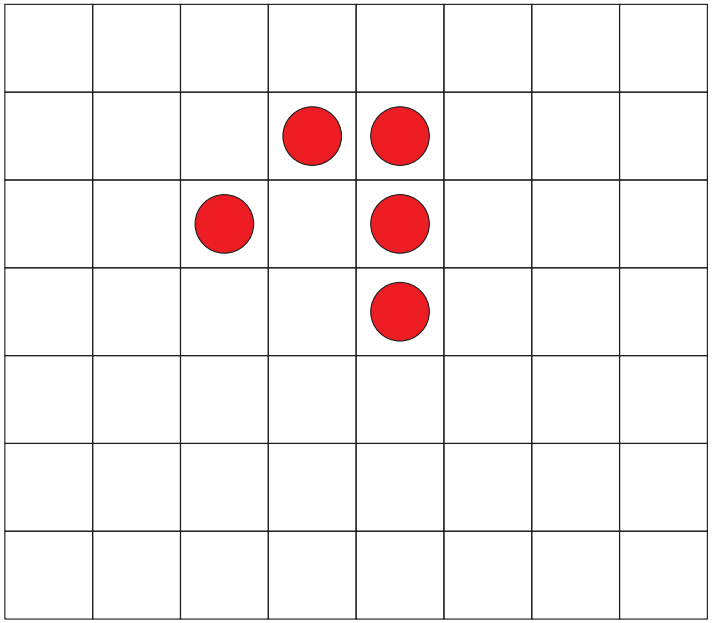
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



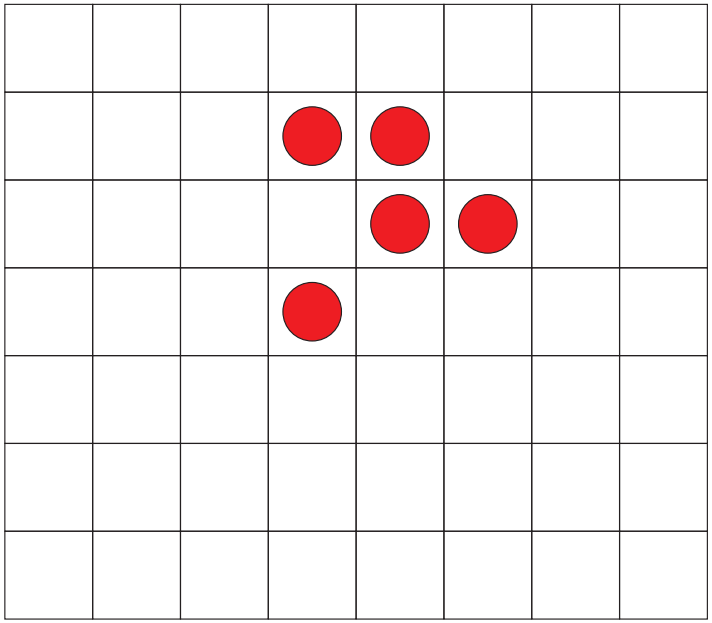
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



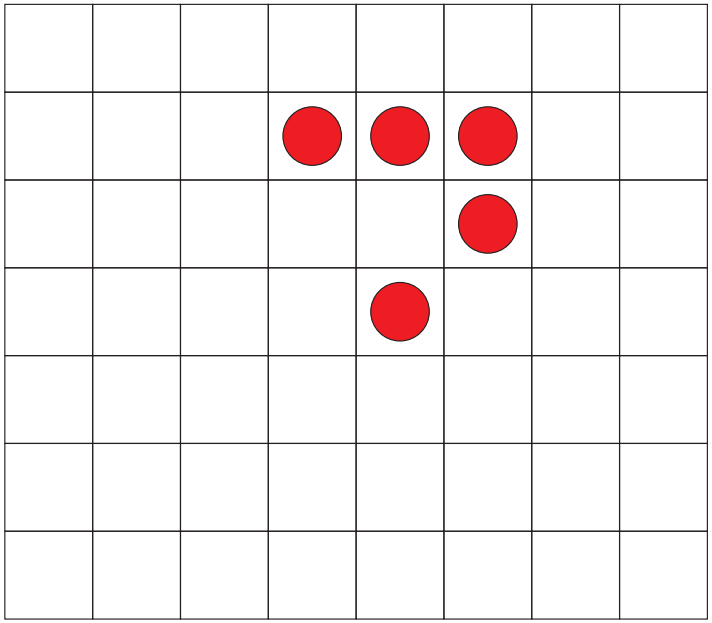
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)

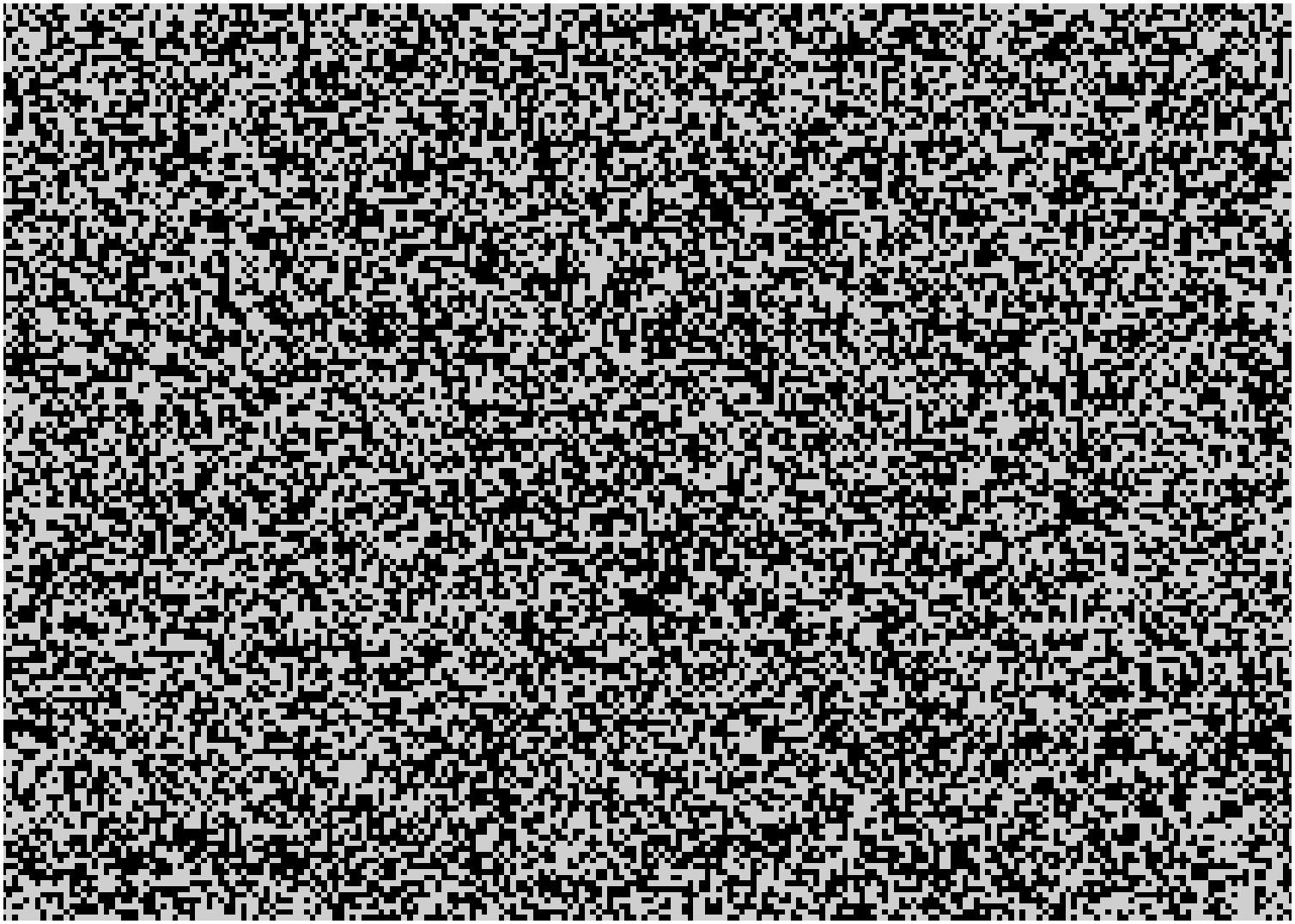


Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**)



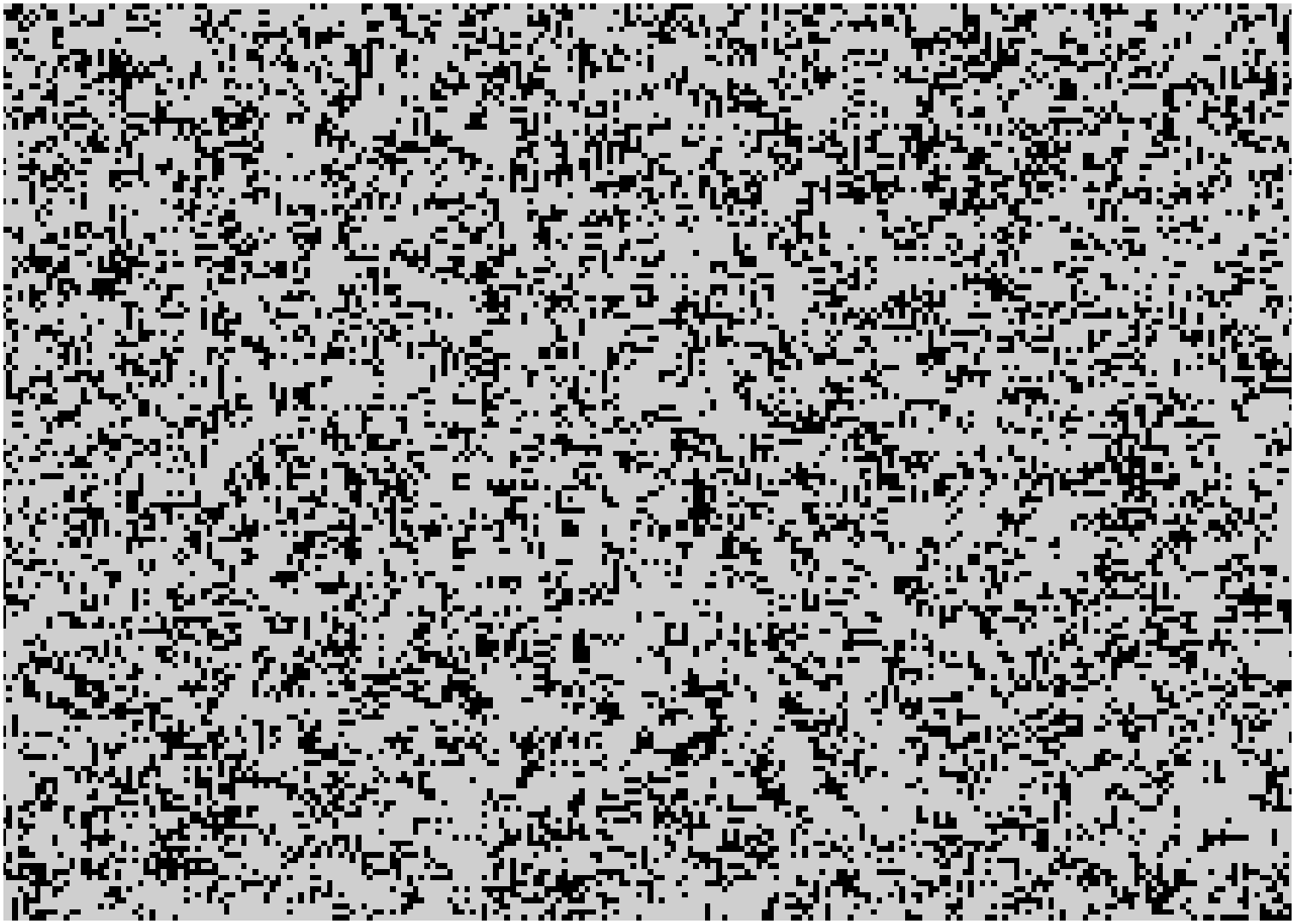
Game of Life was invented in 1970 by John Conway. Since then many interesting "creatures" living in this universe have been identified. These include patterns that remain unchanged (**still life**), oscillate periodically (**oscillators**), glide through space as they oscillate (**spaceships**), emit spaceships (**guns**), etc.

A typical snapshot of a time evolution in Game-of-Life:



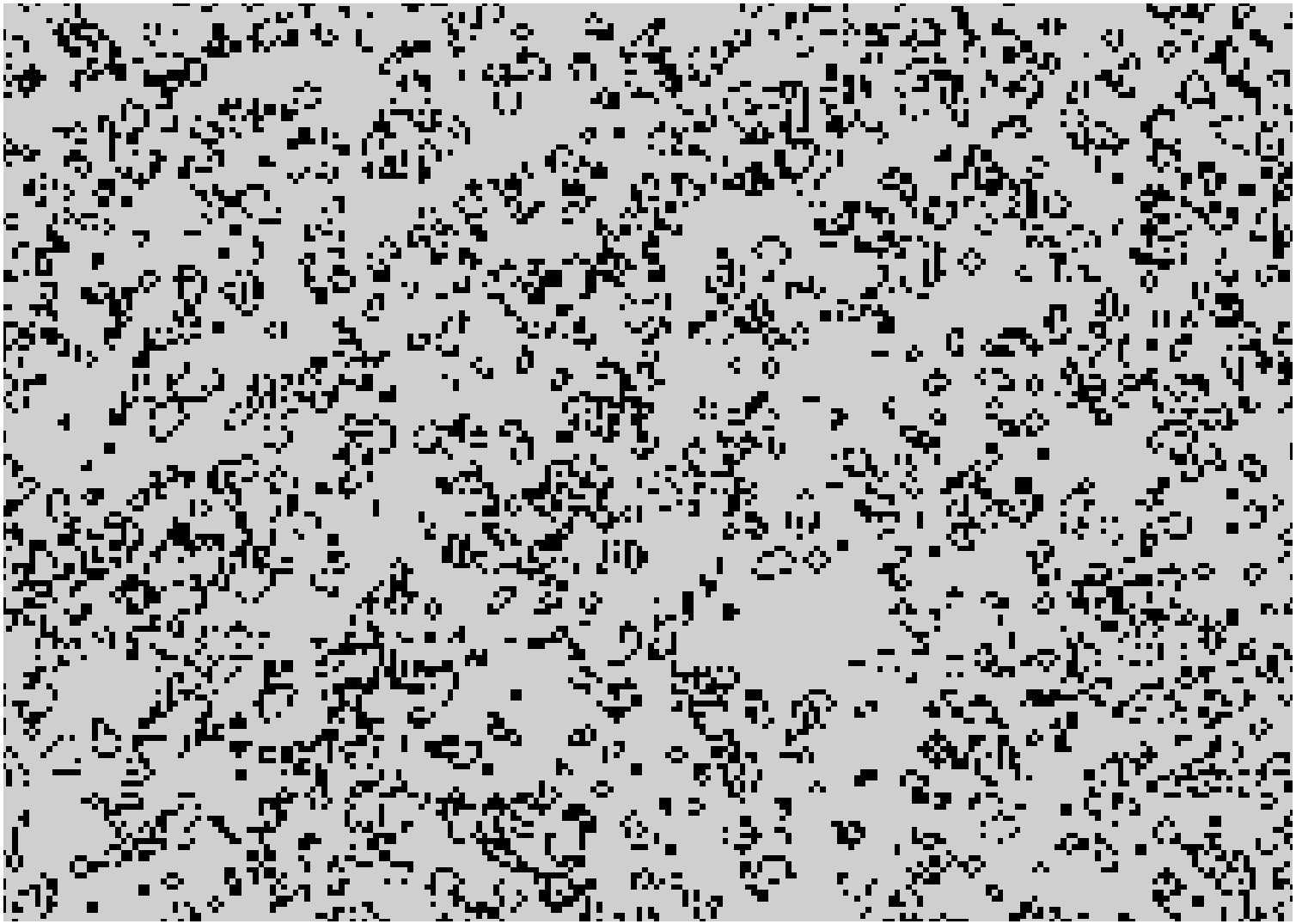
Initial uniformly random configuration.

A typical snapshot of a time evolution in Game-of-Life:



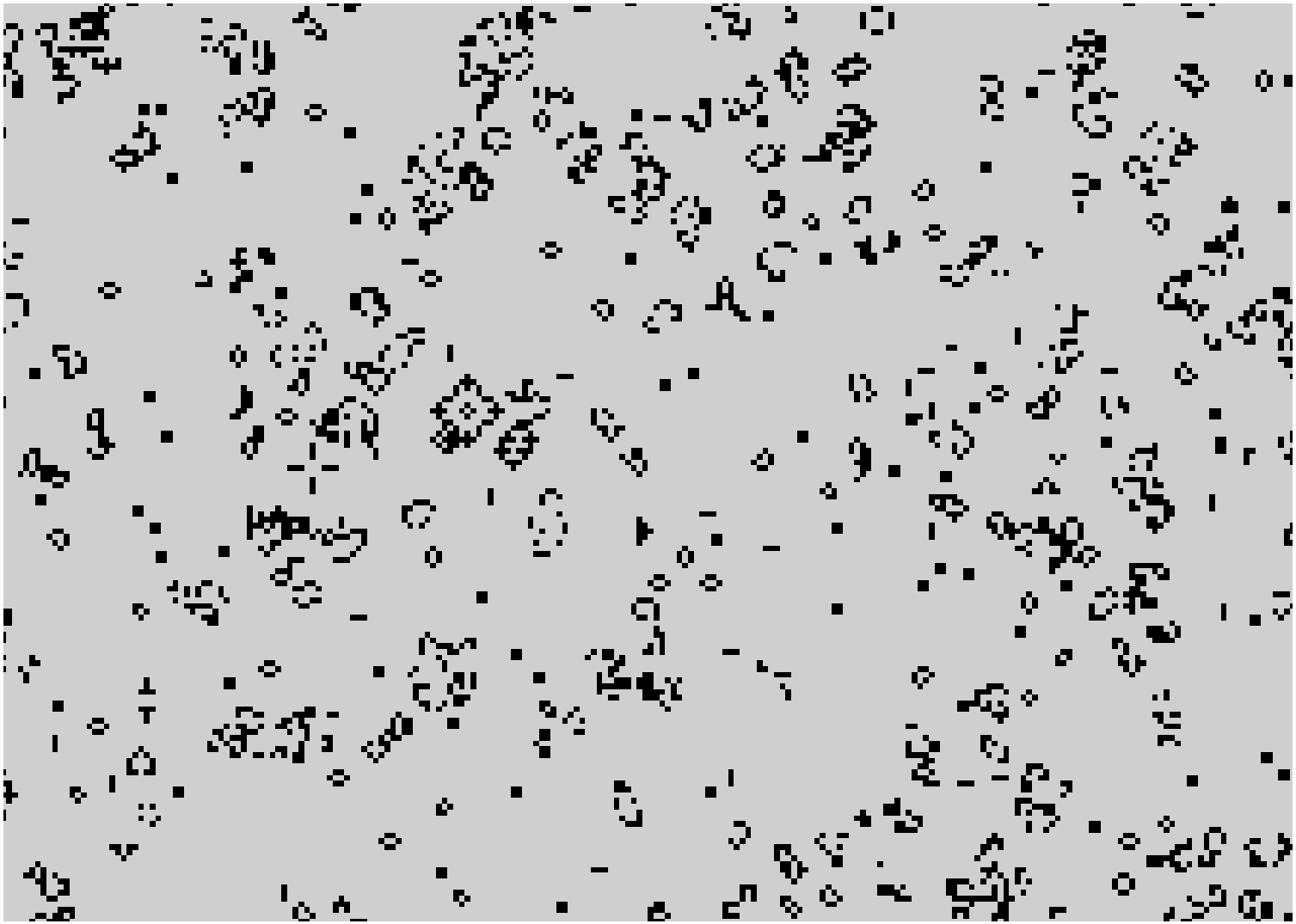
The next generation after all cells applied the update rule.

A typical snapshot of a time evolution in Game-of-Life:



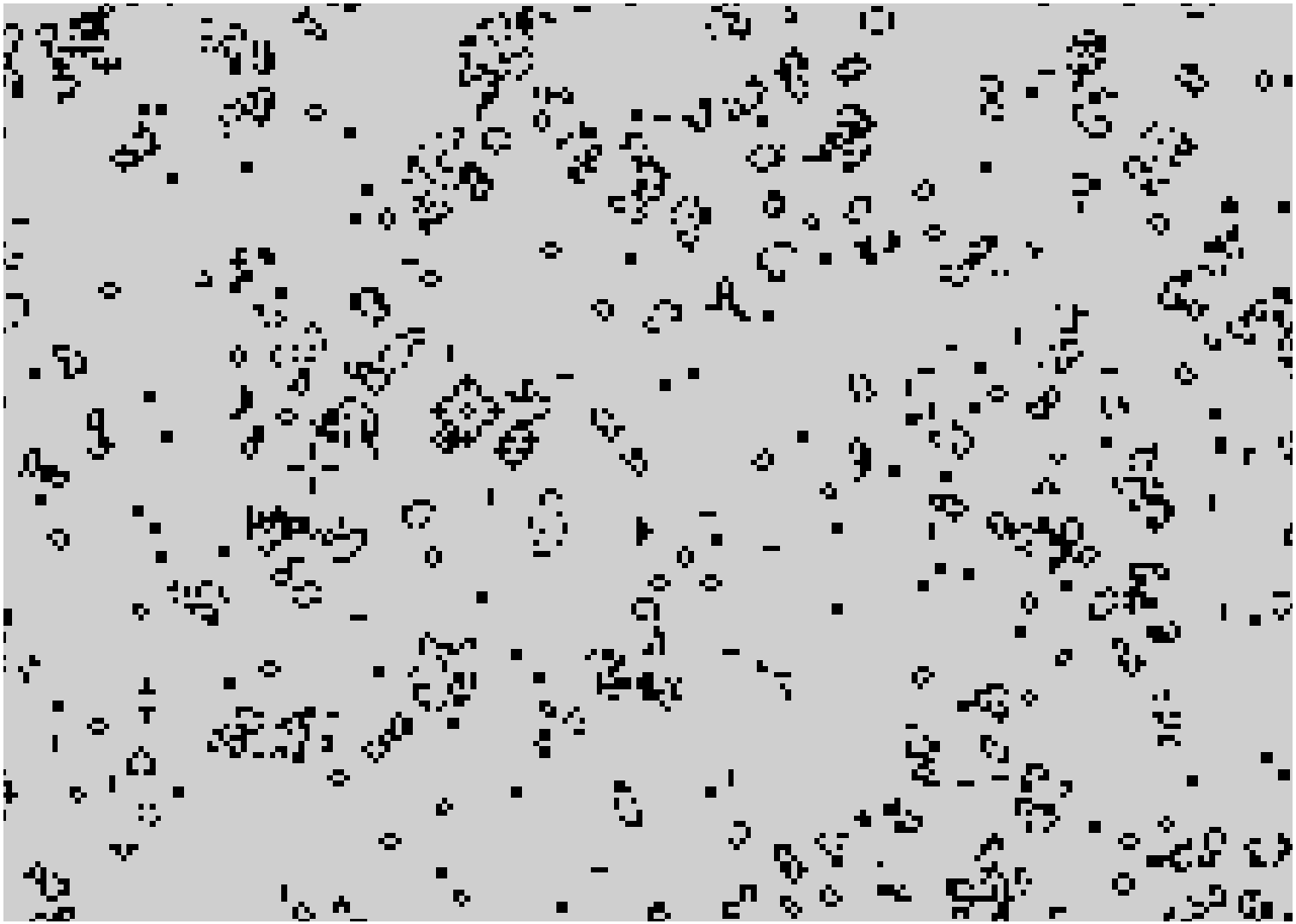
Generation 10

A typical snapshot of a time evolution in Game-of-Life:



Generation 100

A typical snapshot of a time evolution in Game-of-Life:



GOL is a computationally universal two-dimensional CA.

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

- Let d be a positive integer, the **dimension**. A d-dimensional cellular space is the d-dimensional grid $\mathbb{Z}^d$. Elements of $\mathbb{Z}^d$ are called **cells**.

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

- Let d be a positive integer, the **dimension**. A d-dimensional cellular space is the d-dimensional grid $\mathbb{Z}^d$. Elements of $\mathbb{Z}^d$ are called **cells**.
- Let S be a finite **state set**. Elements of S are called **states**.

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

- Let d be a positive integer, the **dimension**. A d-dimensional cellular space is the d-dimensional grid $\mathbb{Z}^d$. Elements of $\mathbb{Z}^d$ are called **cells**.
- Let S be a finite **state set**. Elements of S are called **states**.
- A **configuration** of a d-dimensional CA with state set S is an assignment

$$c : \mathbb{Z}^d \longrightarrow S$$

of states to cells. It is a snapshot of all the states in the system of cells at some moment of time. The state of cell $\vec{n} \in \mathbb{Z}^d$ is $c(\vec{n})$.

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

- Let d be a positive integer, the **dimension**. A d-dimensional cellular space is the d-dimensional grid $\mathbb{Z}^d$. Elements of $\mathbb{Z}^d$ are called **cells**.
- Let S be a finite **state set**. Elements of S are called **states**.
- A **configuration** of a d-dimensional CA with state set S is an assignment

$$c : \mathbb{Z}^d \longrightarrow S$$

  of states to cells. It is a snapshot of all the states in the system of cells at some moment of time. The state of cell $\vec{n} \in \mathbb{Z}^d$ is $c(\vec{n})$.
- The set of all d-dimensional configurations over the state set S is $S^{\mathbb{Z}^d}$. (Uses the mathematical notation B^A for the set of all functions from set A into set B.)

Basic Definitions

Note: abbreviation CA refers to **cellular automata** (plural) or **cellular automaton** (singular).

- Let d be a positive integer, the **dimension**. A d-dimensional cellular space is the d-dimensional grid $\mathbb{Z}^d$. Elements of $\mathbb{Z}^d$ are called **cells**.
- Let S be a finite **state set**. Elements of S are called **states**.
- A **configuration** of a d-dimensional CA with state set S is an assignment

$$c : \mathbb{Z}^d \longrightarrow S$$

  of states to cells. It is a snapshot of all the states in the system of cells at some moment of time. The state of cell $\vec{n} \in \mathbb{Z}^d$ is $c(\vec{n})$.
- The set of all d-dimensional configurations over the state set S is $S^{\mathbb{Z}^d}$. (Uses the mathematical notation B^A for the set of all functions from set A into set B.)
- Most frequently we consider **one**- and **two**-dimensional spaces, in which cases the cells form a line indexed by $\mathbb{Z}$ or an infinite checker board indexed by $\mathbb{Z}^2$. The set of one-dimensional configurations is $S^{\mathbb{Z}}$, the set of functions $\mathbb{Z} \longrightarrow S$.

- A d-dimensional **neighborhood vector** is a tuple

$$N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$$

where each $\vec{n}_i \in \mathbb{Z}^d$ and $\vec{n}_i \neq \vec{n}_j$ for all $i \neq j$. The elements $\vec{n}_i$ specify the relative locations of the neighbors of each cell: Cell $\vec{n} \in \mathbb{Z}^d$ has m **neighbors** $\vec{n} + \vec{n}_i$ for $i = 1, 2, \ldots, m$.

- A d-dimensional **neighborhood vector** is a tuple

$$N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$$

where each $\vec{n}_i \in \mathbb{Z}^d$ and $\vec{n}_i \neq \vec{n}_j$ for all $i \neq j$. The elements $\vec{n}_i$ specify the relative locations of the neighbors of each cell: Cell $\vec{n} \in \mathbb{Z}^d$ has m **neighbors** $\vec{n} + \vec{n}_i$ for $i = 1, 2, \ldots, m$.

- The **local update rule** (or the **local rule**, the **update rule**, or simply the **rule**) of a CA with state set S and size m neighborhood is a function

$$f : S^m \longrightarrow S$$

that specifies the new state of each cell based on the old states of its neighbors: If the neighbors of a cell have states $s_1, s_2, \ldots, s_m$ then the new state of the cell is $f(s_1, s_2, \ldots, s_m)$.

- **Neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$,
- **Local rule** $f : S^m \longrightarrow S$.

- **Neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$,
- **Local rule** $f : S^m \longrightarrow S$.

All cells use the same local rule, and the rule is applied at all cells simultaneously. This causes a global change in the configuration: Configuration c is changed into configuration c' where for all $\vec{n} \in \mathbb{Z}^d$

$$c'(\vec{n}) = f[c(\vec{n} + \vec{n}_1), c(\vec{n} + \vec{n}_2), \ldots, c(\vec{n} + \vec{n}_m)].$$

The transformation $c \mapsto c'$ is the global **transition function**

$$G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$$

of the CA.

- **Neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$,
- **Local rule** $f : S^m \longrightarrow S$.

All cells use the same local rule, and the rule is applied at all cells simultaneously. This causes a global change in the configuration: Configuration c is changed into configuration c' where for all $\vec{n} \in \mathbb{Z}^d$

$$c'(\vec{n}) = f[c(\vec{n} + \vec{n}_1), c(\vec{n} + \vec{n}_2), \ldots, c(\vec{n} + \vec{n}_m)].$$

The transformation $c \mapsto c'$ is the global **transition function**

$$G : S^{\mathbb{Z}^d} \longrightarrow S^{\mathbb{Z}^d}$$

of the CA.

Function G is our main object of study. Functions G that can be defined this way are **CA functions**, or simply cellular automata.

- **Neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$,
- **Local rule** $f : S^m \longrightarrow S$.

Remark: In our notation the neighborhood vector is ordered (=not a set, but a vector) so that the local rule can be expressed simply as a function from m-tuples of states.

- **Neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$,
- **Local rule** $f : S^m \longrightarrow S$.

Remark: In our notation the neighborhood vector is ordered (=not a set, but a vector) so that the local rule can be expressed simply as a function from m-tuples of states.

An **alternative way** is to

- define the neighborhood as a finite subset $N \subseteq \mathbb{Z}^d$,
- take the local rule as a function $f : S^N \longrightarrow S$.

In this notation the domain of f is the set of functions $N \longrightarrow S$, and each such function carries the "order information", i.e., it is known which state each neighbor has.

Transformation G can be iterated, i.e., applied repeatedly, which produces a time evolution

$$c \mapsto G(c) \mapsto G^2(c) \mapsto G^3(c) \mapsto \dots$$

of the system.

Here c is the **initial configuration**, and the sequence

$$\mathcal{O}(c) = c, G(c), G^2(c), G^3(c), \dots$$

is the (forward) **orbit** of c.

By **time** we mean the number of applications of G, so $G^t(c)$ is the configuration at time t.

Transformation G can be iterated, i.e., applied repeatedly, which produces a time evolution

$$c \mapsto G(c) \mapsto G^2(c) \mapsto G^3(c) \mapsto \dots$$

of the system.

Here c is the **initial configuration**, and the sequence

$$\mathcal{O}(c) = c, G(c), G^2(c), G^3(c), \dots$$

is the (forward) **orbit** of c.

By **time** we mean the number of applications of G, so $G^t(c)$ is the configuration at time t.

A **two-way infinite orbit** is a sequence

$$\dots, c_{-2}, c_{-1}, c_0, c_1, c_2, \dots$$

of configurations where $G(c_i) = c_{i+1}$ for all $i \in \mathbb{Z}$.

In summary: To specify a CA one needs to specify the following items:

- the **dimension** $d \in \mathbb{Z}_+$,
- the **finite state set** S,
- the **neighborhood vector** $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$, and
- the **local update rule** $f : S^m \longrightarrow S$.

We therefore formally define the corresponding CA to be the 4-tuple $A = (d, S, N, f)$.

Example: The XOR cellular automaton

Let $d = 1$, $S = \{0, 1\}$, $N = (0, 1)$ and $f : \{0, 1\}^2 \longrightarrow \{0, 1\}$ be

$$f(a, b) = a + b \pmod 2.$$

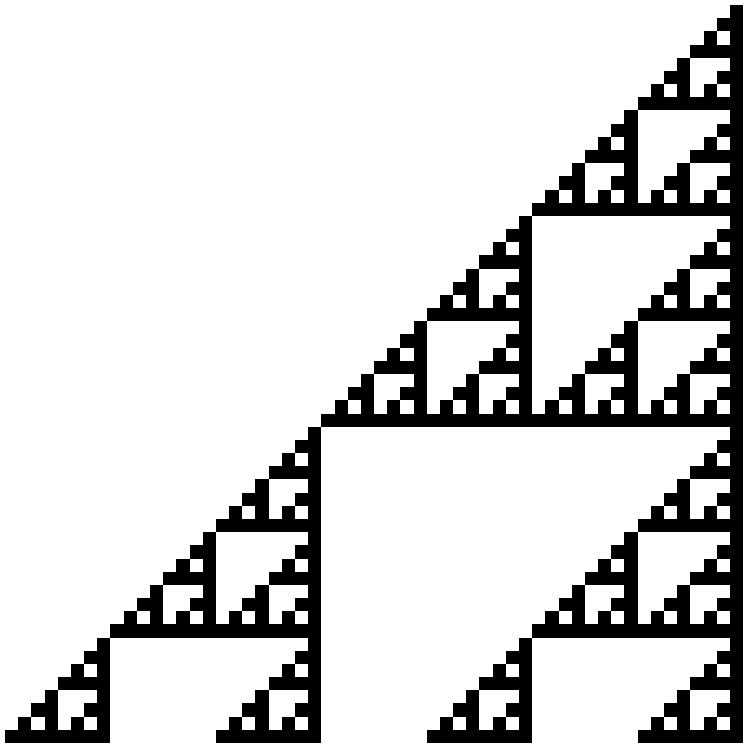
Each cell changes its state by adding the state of its right neighbor to its own old state modulo 2. This is the "exclusive or" (xor) logic operation.

Example: The XOR cellular automaton

Let $d = 1$, $S = \{0, 1\}$, $N = (0, 1)$ and $f : \{0, 1\}^2 \longrightarrow \{0, 1\}$ be

$$f(a, b) = a + b \pmod 2.$$

Each cell changes its state by adding the state of its right neighbor to its own old state modulo 2. This is the "exclusive or" (xor) logic operation.



A **space-time diagram** is a pictorial representation of an orbit.

For one-dimensional CA, the rows of a space-time diagram are consecutive configurations. Time increases downwards. The space-time diagram of the forward orbit of c fills the lower half plane, and the space-time diagrams associated with two-way infinite orbits fill the whole plane $\mathbb{Z}^2$.

A **space-time diagram** is a pictorial representation of an orbit.

For one-dimensional CA, the rows of a space-time diagram are consecutive configurations. Time increases downwards. The space-time diagram of the forward orbit of c fills the lower half plane, and the space-time diagrams associated with two-way infinite orbits fill the whole plane $\mathbb{Z}^2$.

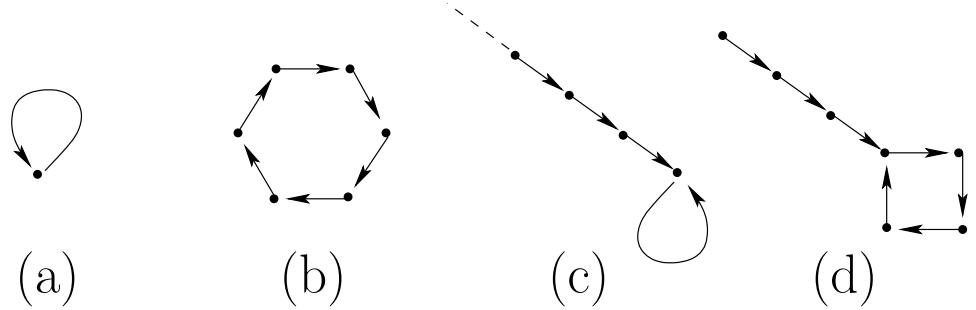
More generally, **a space-time diagram of a d-dimensional CA** is a $(d+1)$-dimensional "drawing" where d dimensions represent space and the additional dimension is used for time.

A configuration c is

- a **fixed point** of G if $G(c) = c$.
- **(temporally) periodic** if $G^t(c) = c$ for some $t \in \mathbb{Z}_+$. Any t satisfying $G^t(c) = c$ is called a **period** of c and the smallest such t is the **least period** of c.
- **eventually fixed** if there is $n \in \mathbb{N}$ such that $G^{n+1}(c) = G^n(c)$, that is, $G^n(c)$ is a fixed point, for some n.
- **eventually (temporally) periodic** if there is $n \in \mathbb{N}$ and $t \in \mathbb{Z}_+$ such that $G^{n+t}(c) = G^n(c)$, that is, $G^n(c)$ is periodic, for some n.

Same terminology for orbits: fixed point orbits, periodic orbits, eventually periodic orbits, etc.

By a **phase space** of G we mean the infinite directed graph whose vertex set is $S^{\mathbb{Z}^d}$ and there is an edge $c \longrightarrow c'$ iff $c' = G(c)$:



(a) a fixed point, (b) a periodic orbit, (c) an eventually fixed orbit and (d) an eventually periodic orbit.

A simple observation: If G and H are CA functions, so is their composition $G \circ H$. $\square$

Neighborhoods

Let

$$N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$$

be a d-dimensional neighborhood vector.

- For any $\vec{n} \in \mathbb{Z}^d$ denote

$$N(\vec{n}) = (\vec{n} + \vec{n}_1, \vec{n} + \vec{n}_2, \ldots, \vec{n} + \vec{n}_m).$$

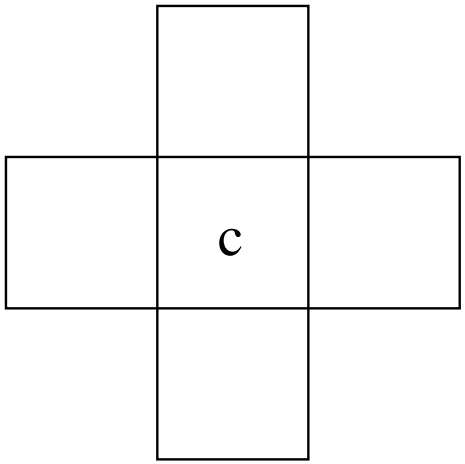
- For any $K \subseteq \mathbb{Z}^d$ denote

$$N(K) = \{\vec{n} + \vec{n}_i \mid \vec{n} \in K \text{ and } i = 1, 2, \ldots, m \}.$$

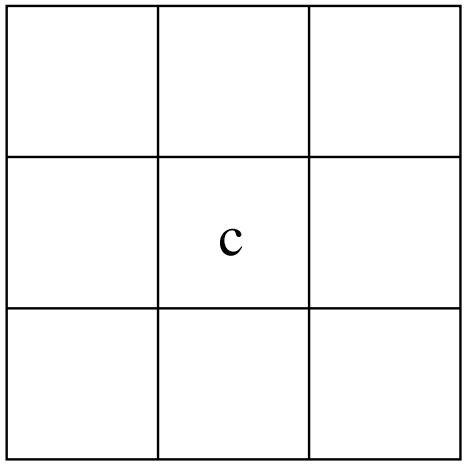
In particular, the difference of $N(\{\vec{n}\})$ and $N(\vec{n})$ is that $N(\{\vec{n}\})$ is unordered (**a set**) while $N(\vec{n})$ is ordered (**a vector**).

Clearly $N = N(\vec{0})$,

In the $d = 2$ case, the (a) **von Neumann** and the (b) **Moore** neighborhoods are often used:

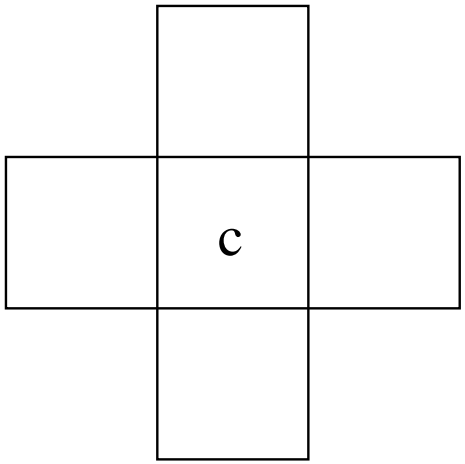


(a)

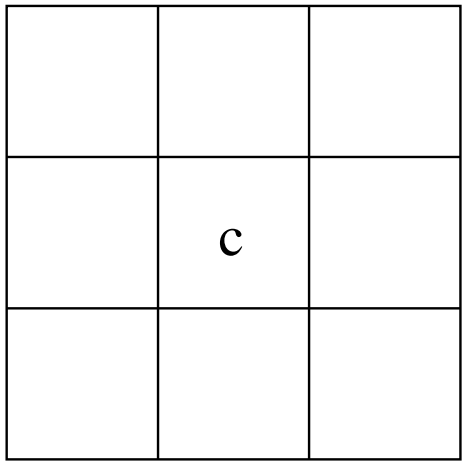


(b)

In the $d = 2$ case, the (a) **von Neumann** and the (b) **Moore** neighborhoods are often used:



(a)



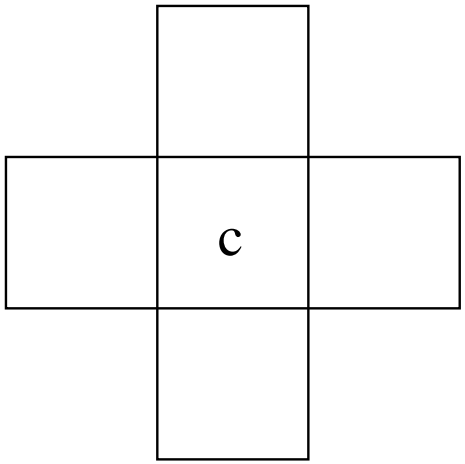
(b)

We generalize the Moore-neighborhood and call the d-dimensional neighborhood M_r^d containing all

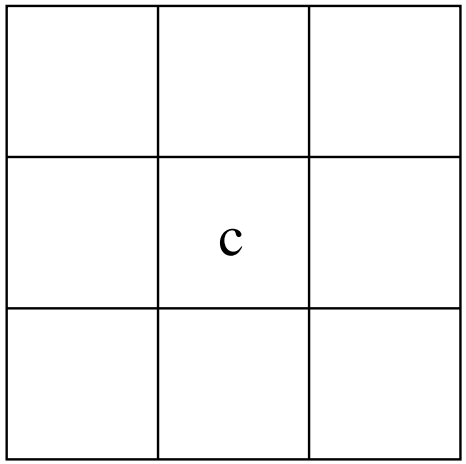
$$(k_1, k_2, \ldots, k_d) \in \mathbb{Z}^d \text{ where } |k_i| \leq r \text{ for all } i = 1, 2, \ldots, d$$

the **radius-r** neighborhood. It contains $(2r + 1)^d$ elements.

In the $d = 2$ case, the (a) **von Neumann** and the (b) **Moore** neighborhoods are often used:



(a)



(b)

We also generalize the von Neumann -neighborhood and call the d-dimensional neighborhood V_r^d consisting of

$$(k_1, k_2, \ldots, k_d) \in \mathbb{Z}^d \text{ where } \sum_{i=1}^{d} |k_i| \leq r$$

the **radius-r von Neumann** neighborhood.

The **radius-$\frac{1}{2}$** neighborhood consists of all $(k_1, k_2, \ldots, k_d) \in \mathbb{Z}^d$ where each $k_i \in \{0, 1\}$, and the radius-$\frac{1}{2}$ von Neumann -neighborhood consists of all $(k_1, k_2, \ldots, k_d) \in \mathbb{Z}^d$ where at most one k_i is 1 and all others are 0.

In the one-dimensional case these are both the same $(0, 1)$. The **XOR** CA uses the radius-$\frac{1}{2}$ neighborhood.

Consider a CA with neighborhood vector $N = (\vec{n}_1, \vec{n}_2, \ldots, \vec{n}_m)$ and local rule $f : S^m \longrightarrow S$.

We call $\vec{n}_j$ a **dummy** neighbor if $f(s_1, \ldots, s_m) = f(t_1, \ldots, t_m)$ whenever $s_i = t_i$ for all $i \neq j$. This means that the the j'th neighbor of a cell has no effect on the next state of that cell, and hence $\vec{n}_j$ can be removed from the neighborhood vector. We obtain an equivalent CA with $m - 1$ neighbors.

By removing all dummy neighbors from any CA we obtain an equivalent CA that has no dummy neighbors. Obviously, this minimal neighborhood is unique:

Proposition. If A and B are equivalent CA (=define the same local function G) and have no dummy neighbors then $A = B$ (up to reordering the neighbors in the neighborhood vector).

Elementary CA

One-dimensional cellular automata with two states and radius-1 neighborhood:

$d = 1$, $S = \{0, 1\}$, $N = (-1, 0, 1)$ and $f : S^3 \longrightarrow S$.

There are 256 elementary CA because the number of different local rules $S^3 \longrightarrow S$ is $2^8 = 256$.

(Some of the 256 elementary rules are identical up to renaming the states or reversing right and left, so the number of essentially different elementary rules is smaller, only 88.)

Elementary rules were extensively studied and empirically classified by S.Wolfram in the 1980's. He introduced a naming scheme that has since become standard: Each elementary rule is specified by an eight bit sequence

$$f(111)\ f(110)\ f(101)\ f(100)\ f(011)\ f(010)\ f(001)\ f(000)$$

where f is the local update rule of the CA. The bit sequence is the binary expansion of an integer in the interval $0 \ldots 255$, called the **Wolfram number** of the CA.

Example. Number 102 in 8-bit binary is

$$01100110$$

so the elementary CA with Wolfram number 102 has the local update rule

$$\begin{array}{llll} f(111) = 0, & f(110) = 1, & f(101) = 1, & f(100) = 0, \\ f(011) = 0, & f(010) = 1, & f(001) = 1, & f(000) = 0, \end{array}$$

This CA is equivalent to the **XOR** CA. (The left neighbor is a dummy neighbor.)

Example.

The 8 bit binary expansion of the decimal number 110 is

$$01101110$$

so the elementary CA with Wolfram number 110 has the local update rule

$$\begin{array}{llll} f(111) = 0, & f(110) = 1, & f(101) = 1, & f(100) = 0, \\ f(011) = 1, & f(010) = 1, & f(001) = 1, & f(000) = 0, \end{array}$$

This CA is known as **rule 110**. It is famous as it is computationally universal (Matthew Cook).

Wolfram experimented in the 80's with elementary CA, and based on empirical observations of their behavior on random initial configurations he classified them into four classes. These are known as **Wolfram classes** of CA. The definitions are not mathematically rigorous.

Wolfram defined the classes as follows:

(W1) Almost all initial configurations lead to the same uniform fixed point configuration,

(W2) Almost all initial configurations lead to a periodically repeating configuration,

(W3) Almost all initial configurations lead to essentially random looking behavior,

(W4) Localized structures with complex interactions emerge.

(W1) Almost all initial configurations lead to the same uniform fixed point configuration,

(W2) Almost all initial configurations lead to a periodically repeating configuration,

(W3) Almost all initial configurations lead to essentially random looking behavior,

(W4) Localized structures with complex interactions emerge.

Class 1: rule 160

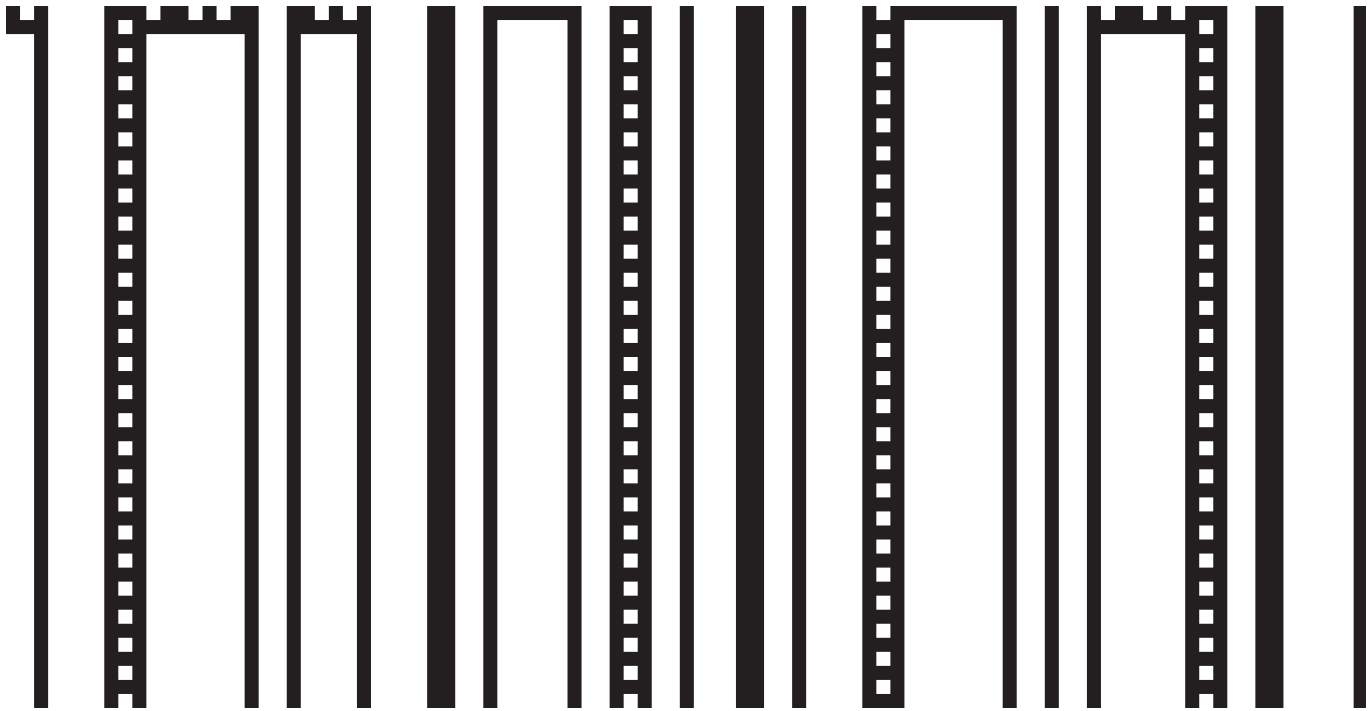
(W1) Almost all initial configurations lead to the same uniform fixed point configuration,

(W2) Almost all initial configurations lead to a periodically repeating configuration,

(W3) Almost all initial configurations lead to essentially random looking behavior,

(W4) Localized structures with complex interactions emerge.

Class 2: rule 108



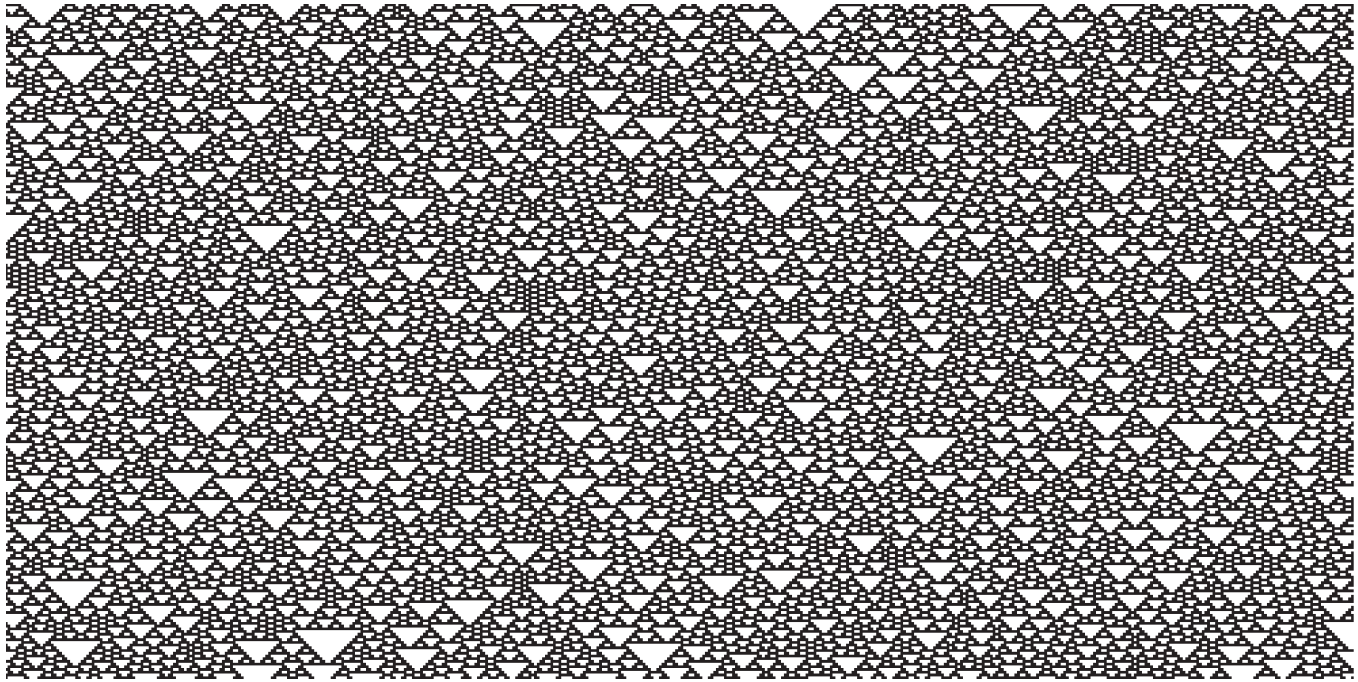
(W1) Almost all initial configurations lead to the same uniform fixed point configuration,

(W2) Almost all initial configurations lead to a periodically repeating configuration,

(W3) Almost all initial configurations lead to essentially random looking behavior,

(W4) Localized structures with complex interactions emerge.

Class 3: rule 126



(W1) Almost all initial configurations lead to the same uniform fixed point configuration,

(W2) Almost all initial configurations lead to a periodically repeating configuration,

(W3) Almost all initial configurations lead to essentially random looking behavior,

(W4) Localized structures with complex interactions emerge.

Class 4: rule 110