A Landscape of Interval Life-like Freezing Cellular Automata

Diego Maldonado
with Eric Goles, Pedro Montealegre, and Martín Ríos-Wilson

Facultad de Ingeniería
Departamento de Ingeniería Informática
Universidad Católica de la Santísima Concepción

13 de julio de 2021

Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

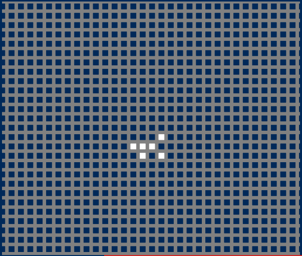
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

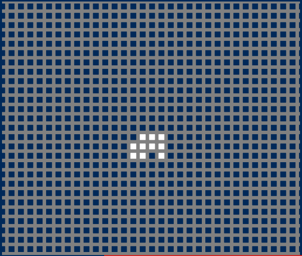
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

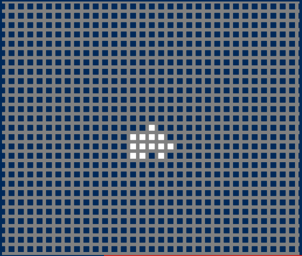
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

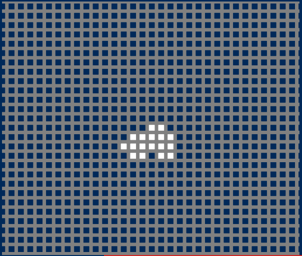
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \ \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

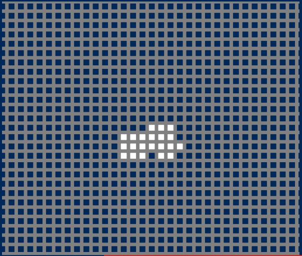
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

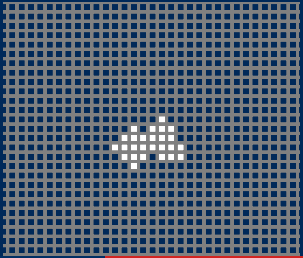
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly 3 } \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

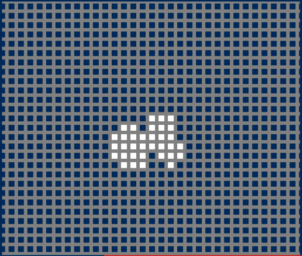
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

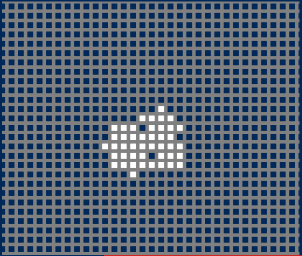
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

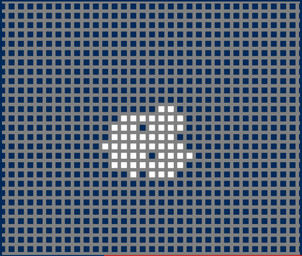
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \ \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

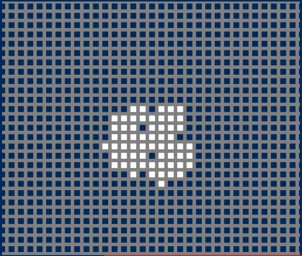
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

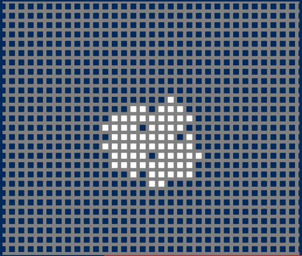
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3\ \blacksquare \\ \text{no change,} & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

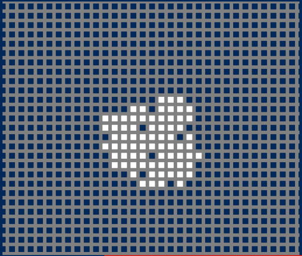
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly 3 } \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

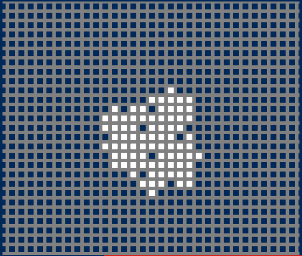
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{□, ■\}$

$f :$  $\mapsto \begin{cases} ■, & \text{if } \exists \text{ exactly } 3\ ■ \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

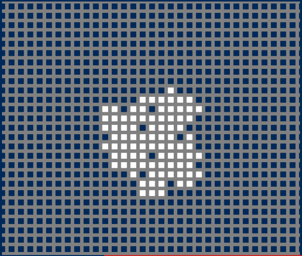
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

Example: Life without death

$Q = \{\square, \blacksquare\}$

f :  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Definition

An two dimensional Cellular Automaton (CA) is a tuple (Q, N, f), where:

- Q is a finite of states.
- $N \subset \mathbb{Z}^2$ finite, called neighborhood.
- $f : Q^N \to Q$ is the local function or rule.
- $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$ is the global function. $F(c)_z = f(c_{N+z})$
- $c \in Q^{\mathbb{Z}^d}$ is a configuration.

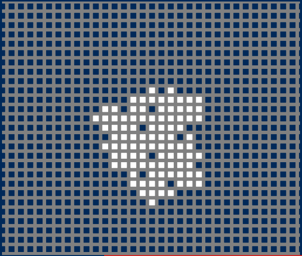
Remark

The temporal dynamic is given by successive iterations of F, i.e., $F^t(c)$

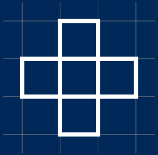
Example: Life without death

$Q = \{\square, \blacksquare\}$

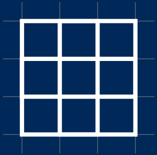
$f :$  $\mapsto \begin{cases} \blacksquare, & \text{if } \exists \text{ exactly } 3 \ \blacksquare \\ \text{no change}, & \text{otherwise.} \end{cases}$



Main neighborhoods

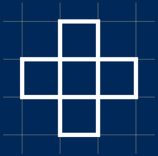


von Neumann neighborhood

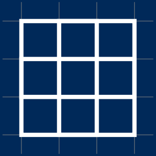


Moore neighborhood

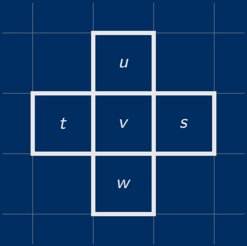
Main neighborhoods



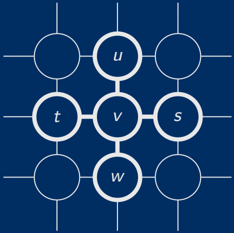
von Neumann neighborhood



Moore neighborhood

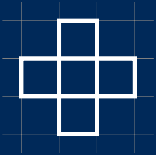


von Neumann CA grid

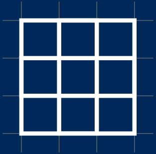


Network interaction graph

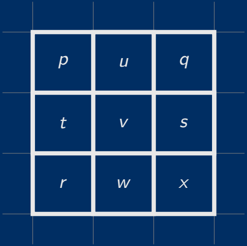
Main neighborhoods



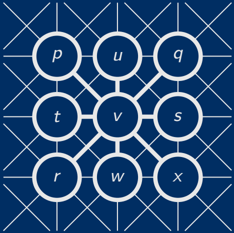
von Neumann neighborhood



Moore neighborhood

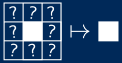


Moore CA grid



Network interaction graph

Definition

A CA with two states is freezing (FCA) if f : 

Definition

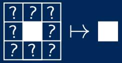
A CA with two states is freezing (FCA) if $f : \begin{array}{|c|c|c|} \hline ? & ? & ? \\ \hline ? & \blacksquare & ? \\ \hline ? & ? & ? \\ \hline \end{array} \mapsto \blacksquare$

Definition

Given a configuration c, a cell $z \in \mathbb{Z}^2$ is unstable if

$$\exists T : F^T(c)_z \neq c_z$$

Definition

A CA with two states is freezing (FCA) if f : 

Definition

Given a configuration c, a cell $z \in \mathbb{Z}^2$ is unstable if

$$\exists T : F^T(c)_z \neq c_z$$

Definition ($\texttt{Unstability}_F$)

F is a FCA.

INPUT: A $n \times n$-periodic configuration c and a cell z.

QUESTION: Does there exist a time $T > 0$ such that $F^T(c)_z \neq c_z$?

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Theorem

$\mathtt{Unstability}_F$ is in P

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Theorem

$\mathtt{Unstability}_F$ is in P

Definition (NC class)

NC is the set of decision problems solvable in poly-log time in a parallel machine with polynomial processors.

Theorem

$NC \subseteq P$.

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Theorem

$\mathtt{Unstability}_F$ is in P

Definition (NC class)

NC is the set of decision problems solvable in poly-log time in a parallel machine with polynomial processors.

Theorem

$NC \subseteq P$. **Open question:** $NC = P$?

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Theorem

$\mathtt{Unstability}_F$ is in P

Definition (NC class)

NC is the set of decision problems solvable in poly-log time in a parallel machine with polynomial processors.

Theorem

$NC \subseteq P$. **Open question:** *$NC = P$?*

Definition (P-completeness)

A problem $p \in P$ is P-complete if $\forall q \in P \; q \preccurlyeq_* p$.

Definition (P class)

P is the set of decision problems solvable in polynomial time in a sequential machine.

Theorem

$\mathtt{Unstability}_F$ is in P

Definition (NC class)

NC is the set of decision problems solvable in poly-log time in a parallel machine with polynomial processors.

Theorem

$NC \subseteq P$. **Open question:** $NC = P$?

Definition (P-completeness)

A problem $p \in P$ is P-complete if $\forall q \in P \; q \preccurlyeq_* p$.

Remark

- If there exists P-complete problem in NC, then $NC = P$.
- P-complete are consider intrinsically sequential problems.

Questions

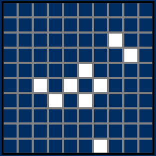
How complex are FCA in restricted environments?
How increase the complexity if the neighborhood increase?

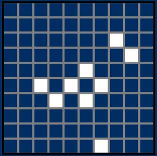
Approach

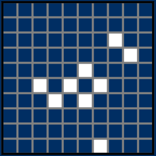
Complex behaviors are hard to compute.
For that:

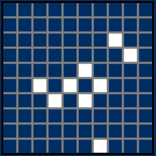
- To Study FCA two states, von Neumann neighborhood and totalistic. (Previous work)
- To Study FCA two states, Moore neighborhood and Life-like. (This exploration)

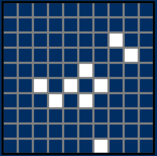
A FCA totalistic changes to ■ if has some numbers of ■ near. We call them using these numbers.

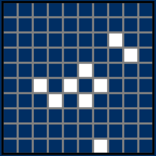

Rule 4.

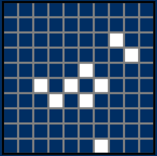

Rule 3.

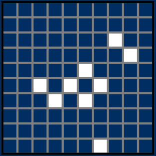

Rule 34.

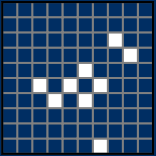

Rule 2.

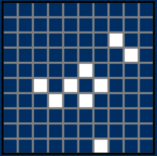

Rule 24.

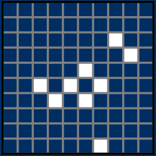

Rule 23.

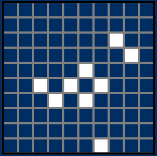

Rule 234.

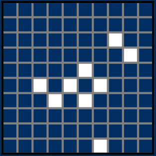

Rule 1.

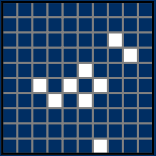

Rule 14.

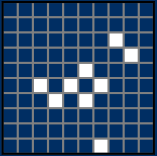

Rule 13.


Rule 134.


Rule 12.

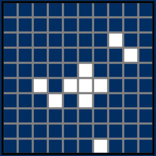

Rule 124.

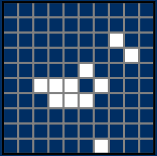

Rule 123.

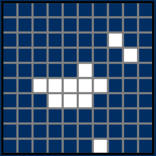

Rule 1234.

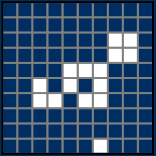
We can classify these TFCA in 5 groups: Fractals, trivial, algebraic, threshold-like, P-complete

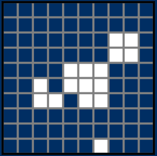
A FCA totalistic changes to ■ if has some numbers of ■ near. We call them using these numbers.

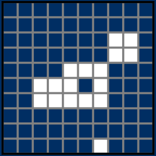

Rule 4.

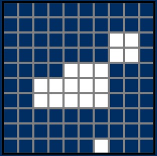

Rule 3.

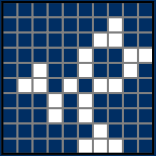

Rule 34.

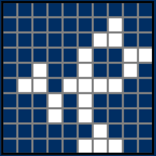

Rule 2.

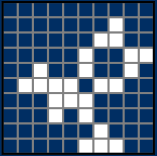

Rule 24.

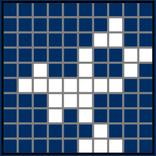

Rule 23.

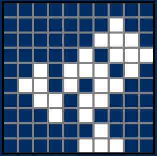

Rule 234.

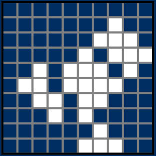

Rule 1.

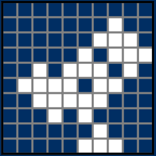

Rule 14.

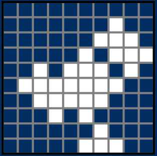

Rule 13.


Rule 134.


Rule 12.

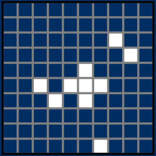

Rule 124.

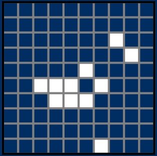

Rule 123.

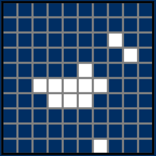

Rule 1234.

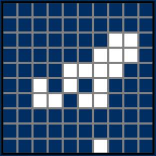
We can classify these TFCA in 5 groups: Fractals, trivial, algebraic, threshold-like, P-complete

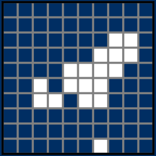
A FCA totalistic changes to ■ if has some numbers of ■ near. We call them using these numbers.

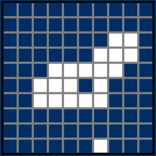

Rule 4.

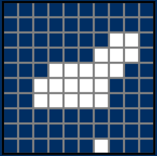

Rule 3.

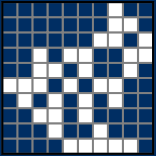

Rule 34.

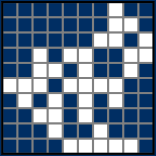

Rule 2.

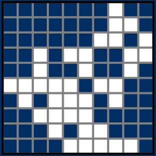

Rule 24.

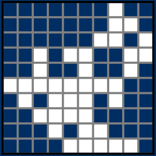

Rule 23.

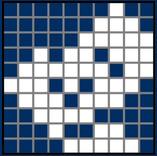

Rule 234.

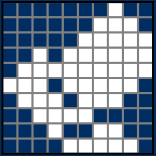

Rule 1.

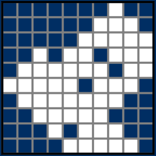

Rule 14.

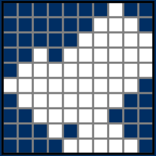

Rule 13.


Rule 134.


Rule 12.

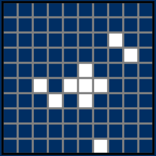

Rule 124.

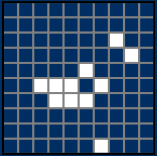

Rule 123.

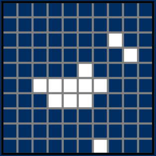

Rule 1234.

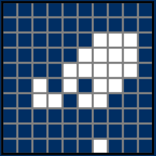
We can classify these TFCA in 5 groups: Fractals, trivial, algebraic, threshold-like, P-complete

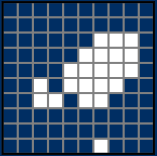
A FCA totalistic changes to ■ if has some numbers of ■ near. We call them using these numbers.

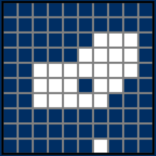

Rule 4.

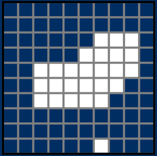

Rule 3.

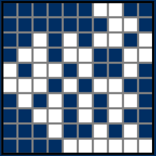

Rule 34.

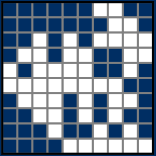

Rule 2.

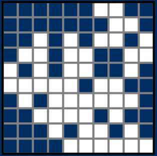

Rule 24.

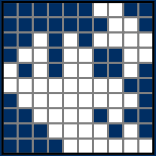

Rule 23.

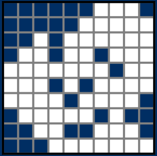

Rule 234.

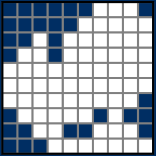

Rule 1.

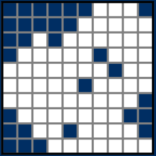

Rule 14.

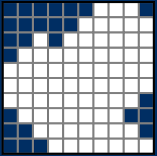

Rule 13.


Rule 134.


Rule 12.

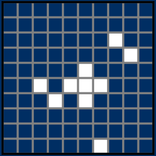

Rule 124.

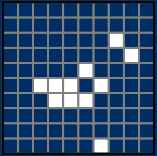

Rule 123.

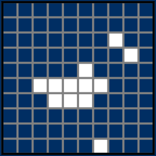

Rule 1234.

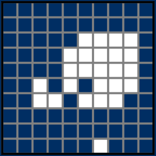
We can classify these TFCA in 5 groups: Fractals, trivial, algebraic, threshold-like, P-complete

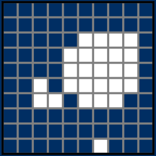
A FCA totalistic changes to ■ if has some numbers of ■ near. We call them using these numbers.

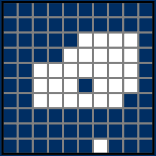

Rule 4.

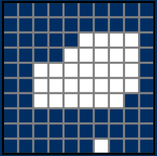

Rule 3.

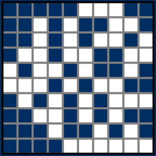

Rule 34.

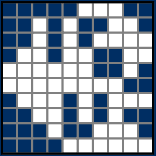

Rule 2.

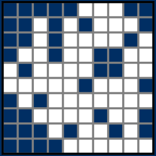

Rule 24.

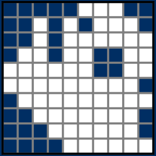

Rule 23.

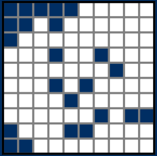

Rule 234.

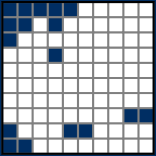

Rule 1.

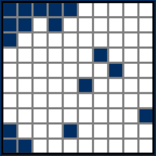

Rule 14.

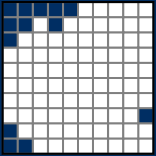

Rule 13.


Rule 134.

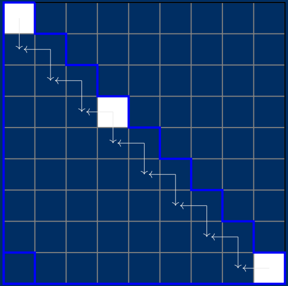

Rule 12.


Rule 124.


Rule 123.


Rule 1234.

We can classify these TFCA in 5 groups: Fractals, trivial, algebraic, threshold-like, P-complete

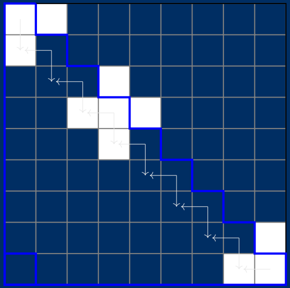


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

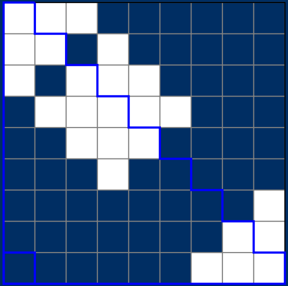


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in *NC*.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.



Input: Finite configuration in a square of $n \times n$.

Find smallest blue square s.t. boundary have a cell 1.

if not exists blue square **then**

  return no change

end if

Compute red cells

Compute green cells

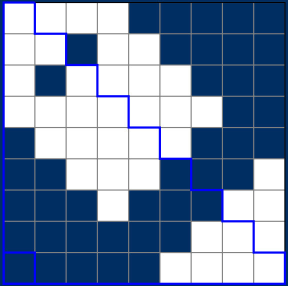
Compute center cell

Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

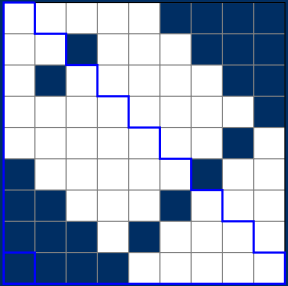


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.



Input: Finite configuration in a square of $n \times n$.

Find smallest blue square s.t. boundary have a cell 1.

if not exists blue square **then**

return no change

end if

Compute red cells

Compute green cells

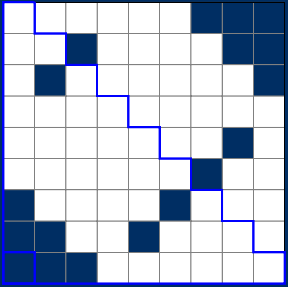
Compute center cell

Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.



Input: Finite configuration in a square of $n \times n$.

Find smallest blue square s.t. boundary have a cell 1.

if not exists blue square **then**

return no change

end if

Compute red cells

Compute green cells

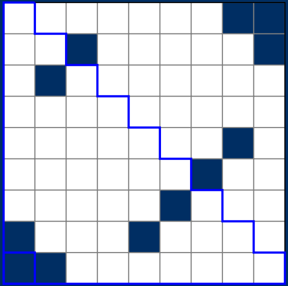
Compute center cell

Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.



Input: Finite configuration in a square of $n \times n$.

Find smallest blue square s.t. boundary have a cell 1.

if not exists blue square **then**

return no change

end if

Compute red cells

Compute green cells

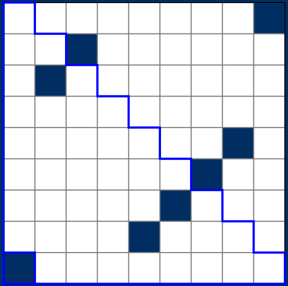
Compute center cell

Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

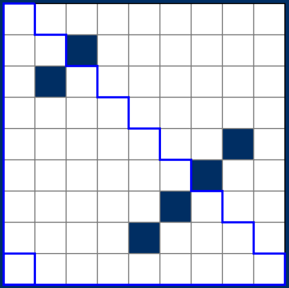


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

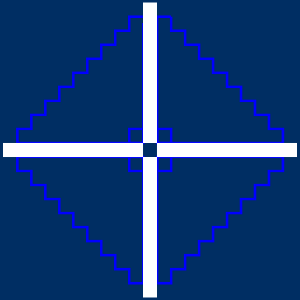


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

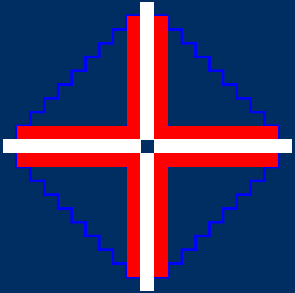


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

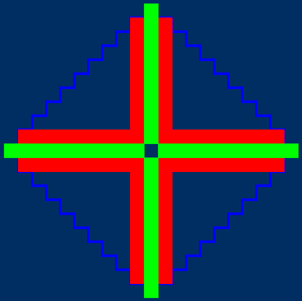


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

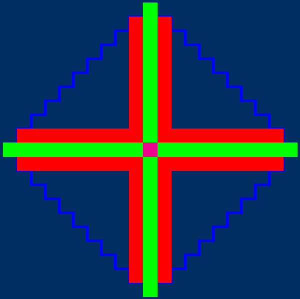


Input: Finite configuration in a square of $n \times n$.
Find smallest blue square s.t. boundary have a cell 1.
if not exists blue square **then**
return no change
end if
Compute red cells
Compute green cells
Compute center cell
Output: Value in the center cell.

Computable in NC.

Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.



Input: Finite configuration in a square of $n \times n$.

Find smallest blue square s.t. boundary have a cell 1.

if not exists blue square **then**

return no change

end if

Compute red cells

Compute green cells

Compute center cell

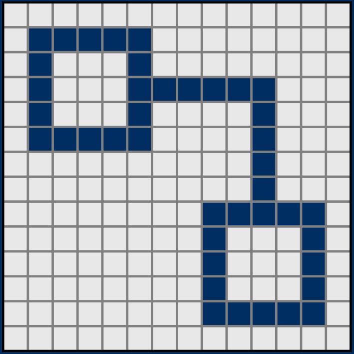
Output: Value in the center cell.

Computable in NC.

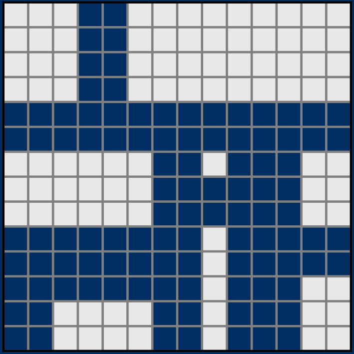
Theorem (Prefix sum problem)

The problem to "sum" n elements is computable in NC.

Strict majority (Rule 34)



Non Strict majority (Rule 234)



Theorem (Goles, Montealegre, Todinca 2013)

A cells is stable iff it is in cycles or paths between cycles, then this structure is computable in NC.

Theorem

A cells is stable iff it is in a 3-connected component, then this structure is computable in NC.

Theorem

The problem to compute a k-connected component is computable in NC.

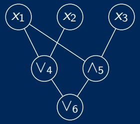
P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

Proof (idea).



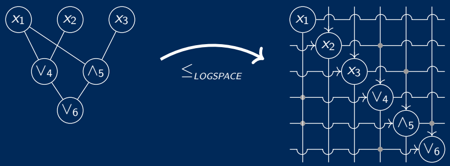
P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

Proof (idea).



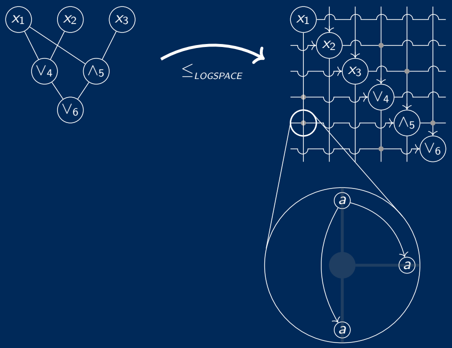
P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

Proof (idea).



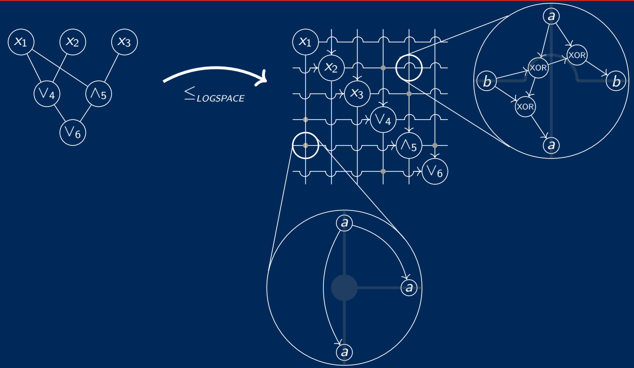
P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

Proof (idea).



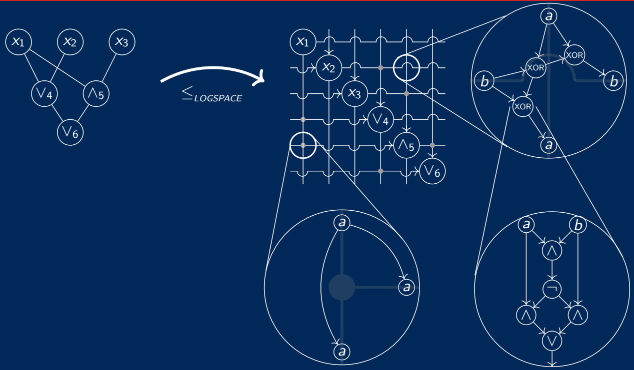
P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

Proof (idea).



P-Complete problems:

- Circuit value problem
- Planar Circuit value problem
- Monotone Circuit value problem

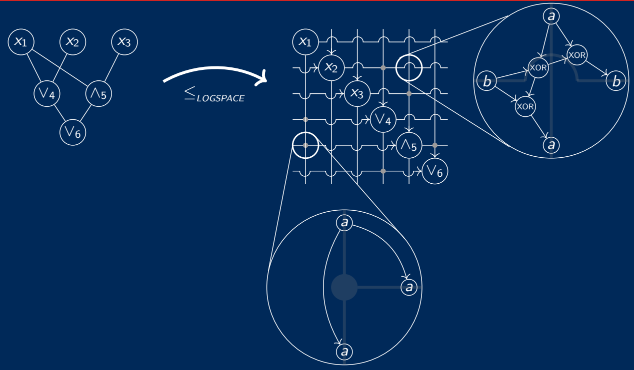
Theorem (Goldschlager,1977)

Planar Circuit value is P-complete

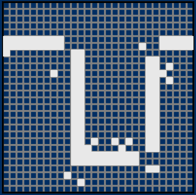
Corollary

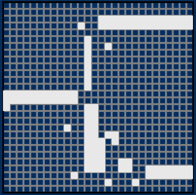
Monotone-XOR Planar Circuit value is P-complete.

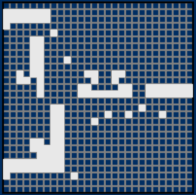
Proof (idea).

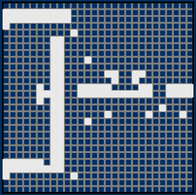


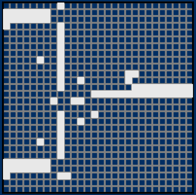
P-completeness *à la* Banks.


Wires and turns


Duplicator


AND gate

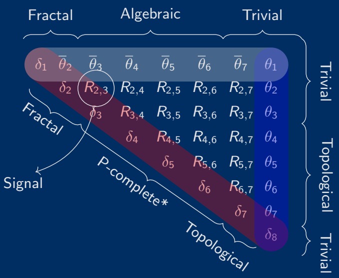

OR gate

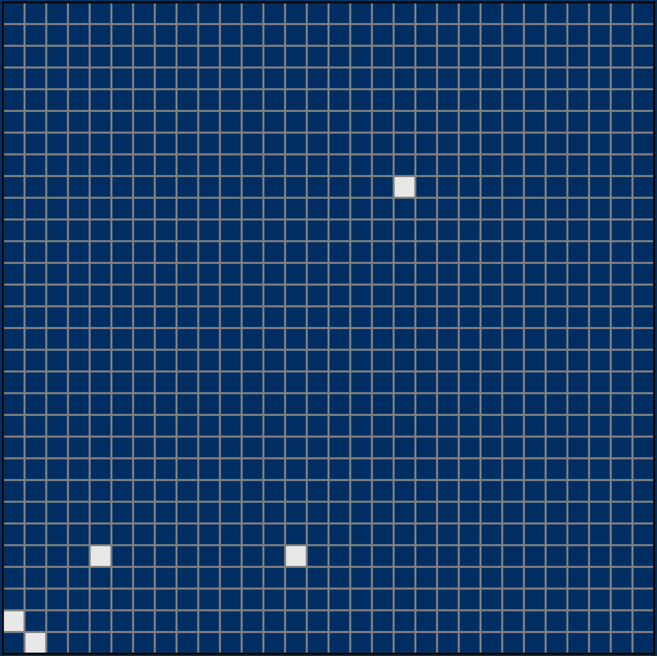

XOR gate

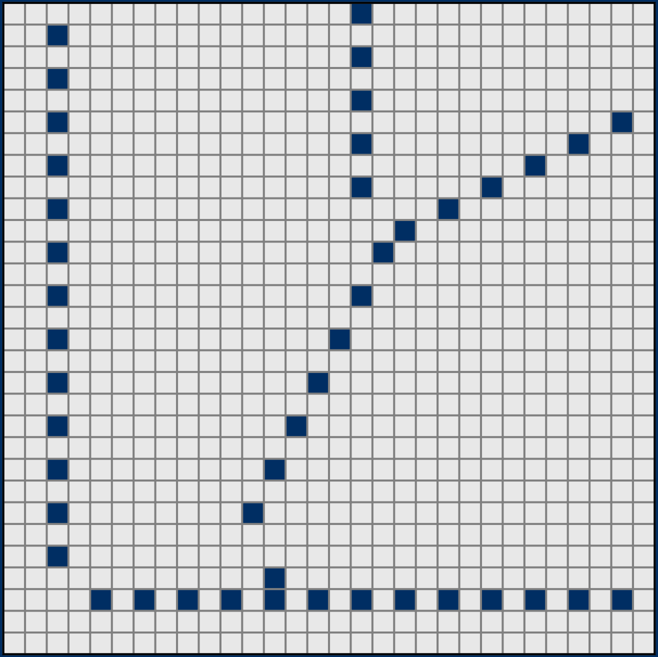
- We are studying Interval Life-Like Cellular Automaton FCA with 2-states and Moore neighborhood. i.e. for $0 \leq a < b \leq 8$:

$$f(c_{(i,j)+N}) = 1$$
$$\Updownarrow$$
$$c_{(i,j)} = 1 \vee c_{(i,j)} = 0 \wedge a \leq \sum_{v \in (i,j)+N} c_v \leq b$$

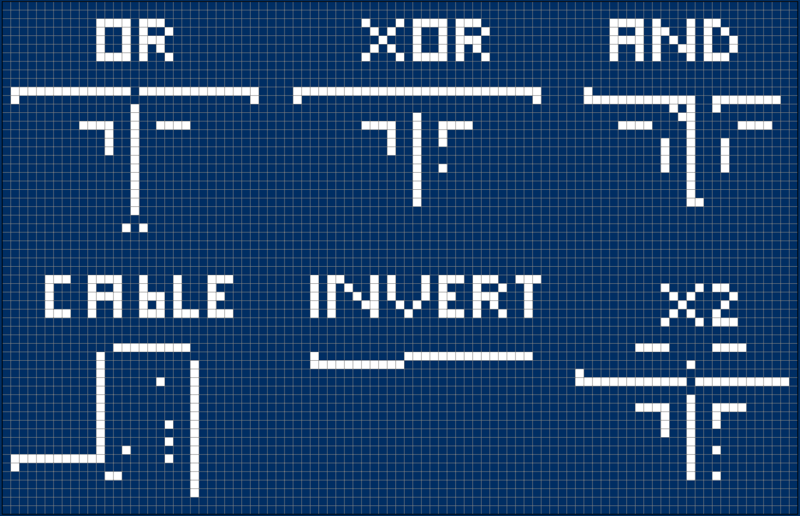
- We will call these CA $R_{i,j}$. e.g. the Life without Death is $R_{3,3}$.
- ILLFCA include threshold CA with threshold, e.g. the non-strict majority is θ_4.
- We define the anti-threshold ($\overline{\theta}_k$) as the ILLFCA that remains inactive iff the total of active neighbors is $\geq k$ or 0, e.g. $R_{1,3} = \overline{\theta}_4$.
- We define δ_k rule as the rule actives a cell iff there is exactly k active neighbors, e.g. the Life without Death is δ_3.



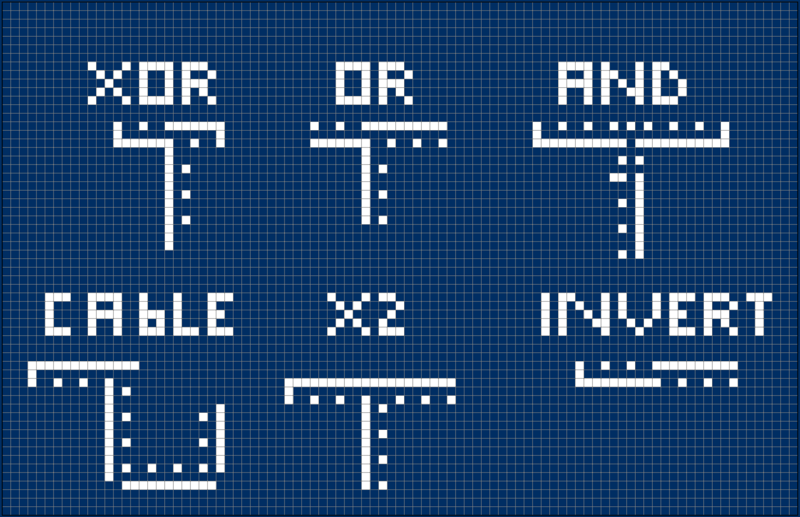




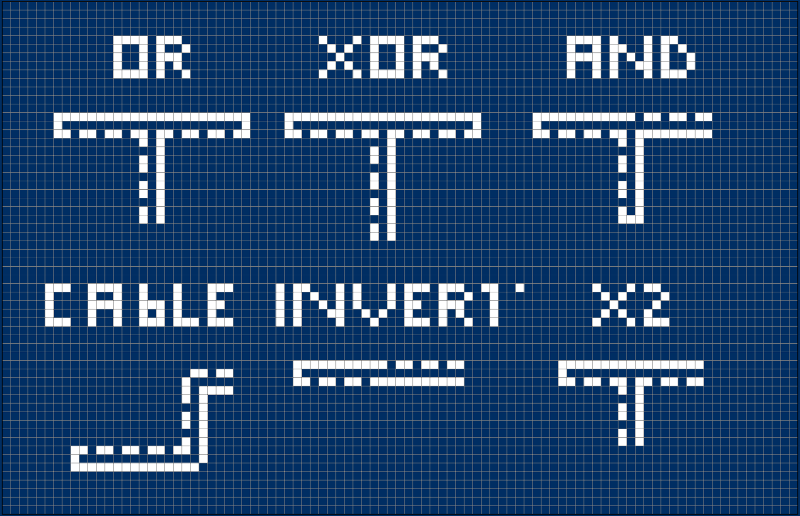
P-complete δ_4



P-complete δ_5

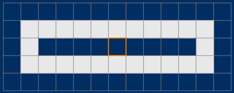


P-complete δ_6

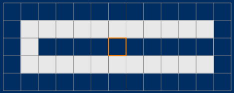


P-complete δ_7

A cell changes if has 7 active neighbors, or has single signal arriving alone. 2 signals arriving simultaneously stabilizes the cell.



Two signals stabilizing a cell.



No stable configuration.

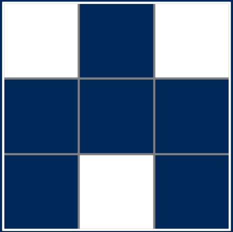
Threshold θ_1

If there is 1 active cell, then every cell changes.

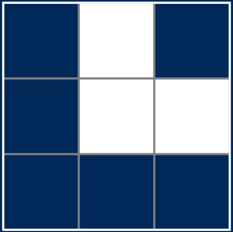
Threshold θ_2

If there are 2 cells sharing a neighborhood, then every cell changes.

Threshold θ_3



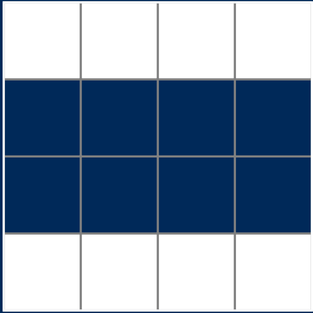
Spreading configuration.



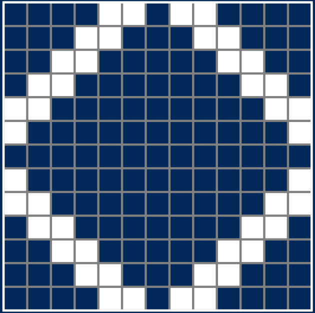
Non-spreading configuration.

Threshold θ_4 (non strict majority)

A cell is stable if it has 5 stable neighbors.



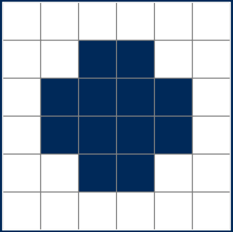
Stable configuration in a 5-connected component.



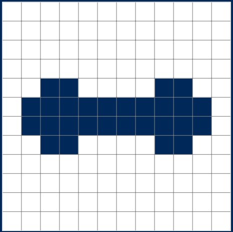
Stable configuration in a non 5-connected component.

Threshold θ_5 (strict majority)

A cell is stable if it has 4 stable neighbors.



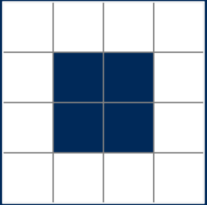
Bounded stable configuration.



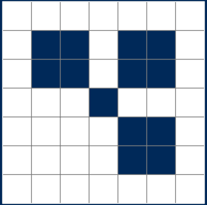
Two bounded stable configuration stabilizing a non-stabilized pattern.

Threshold θ_6

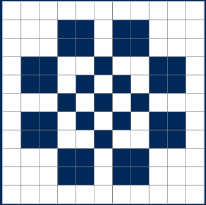
A cell is stable if it has 3 stable neighbors.



Bounded stable configuration.



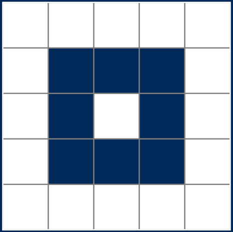
3 bounded stable configuration stabilizing a center cell.



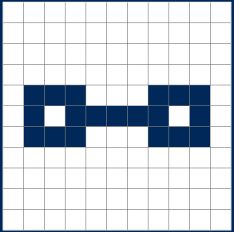
8 bounded stable configuration stabilizing a non-stabilized pattern.

Threshold θ_7

A cell is stable if it has 2 stable neighbors.



Bounded stable configuration.



2 bounded stable configuration stabilizing a path.

- A new behavior appears, a rule sending signals, $R_{2,3}$.
- A P-complete rule appears outside the diagonal, $R_{5,6}$.
- Study how deep in the triangle there is complexity.
- Study what happens to the fractal rules.
- Extend the results of the Threshold rules to arbitrary large neighborhoods.

- A new behavior appears, a rule sending signals, $R_{2,3}$.
- A P-complete rule appears outside the diagonal, $R_{5,6}$.
- Study how deep in the triangle there is complexity.
- Study what happens to the fractal rules.
- Extend the results of the Threshold rules to arbitrary large neighborhoods.

Thanks for your attention !!!