Introduction à l'informatique
Automates cellulaires

Portail René Descartes

Luminy

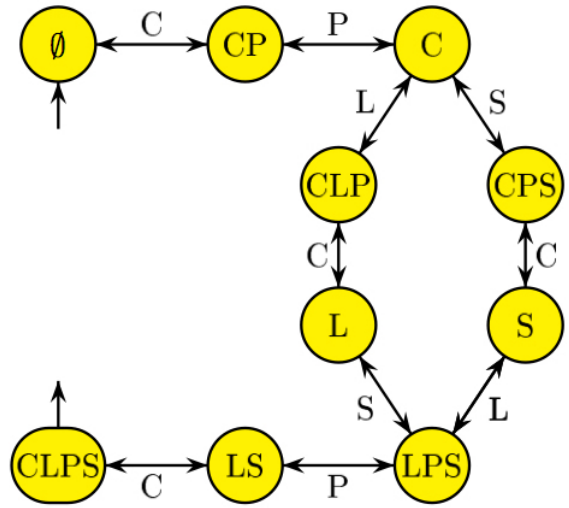
Université d'Aix-Marseille

Plan du cours

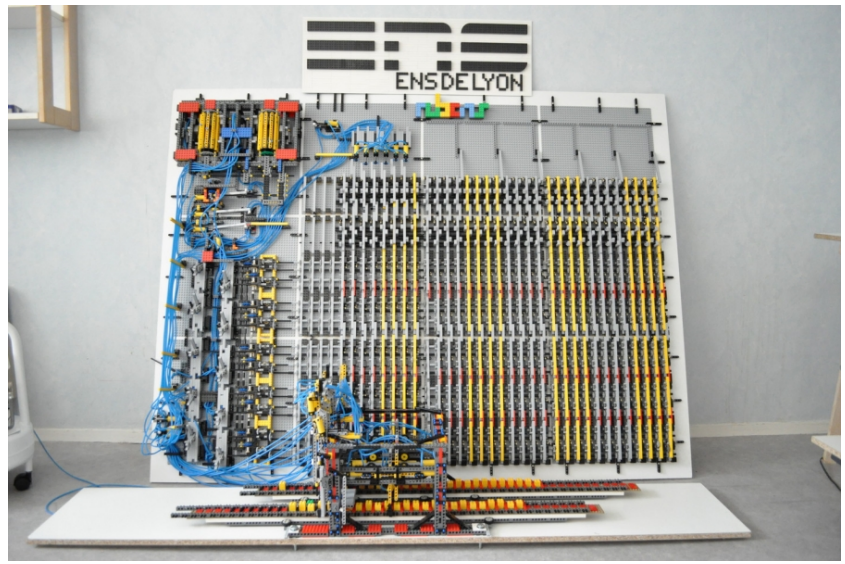
Plan du cours

Machines de Turing

- Automates finis
(chèvre, loup, salade)



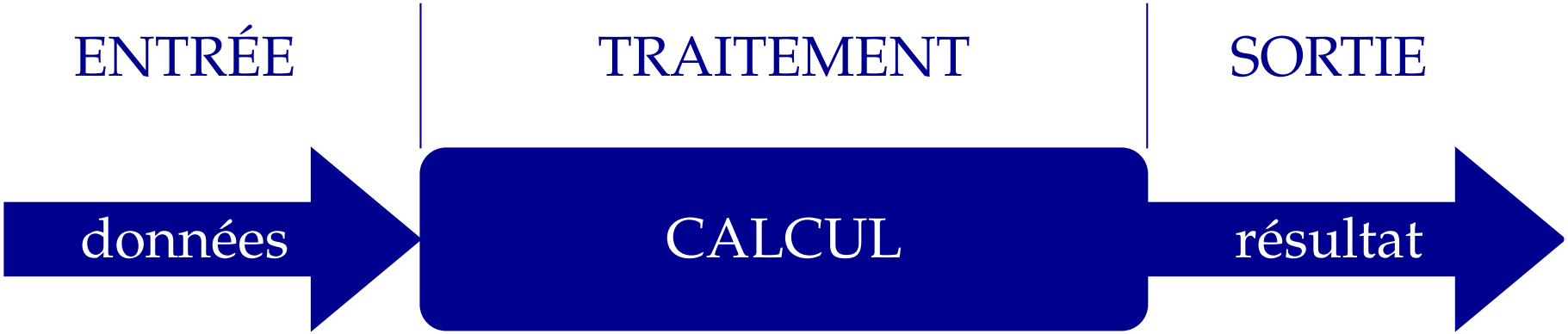
- Machines de Turing
(une machine en lego)



Peut-on tout calculer ?

- Étant donné un problème bien formulé d'un point de vue mathématique...
- avec des entrées sous la forme de séquences de symboles et des sorties oui ou non...
- existe-t-il toujours une machine de Turing (ou un algorithme) qui peut résoudre le problème ?

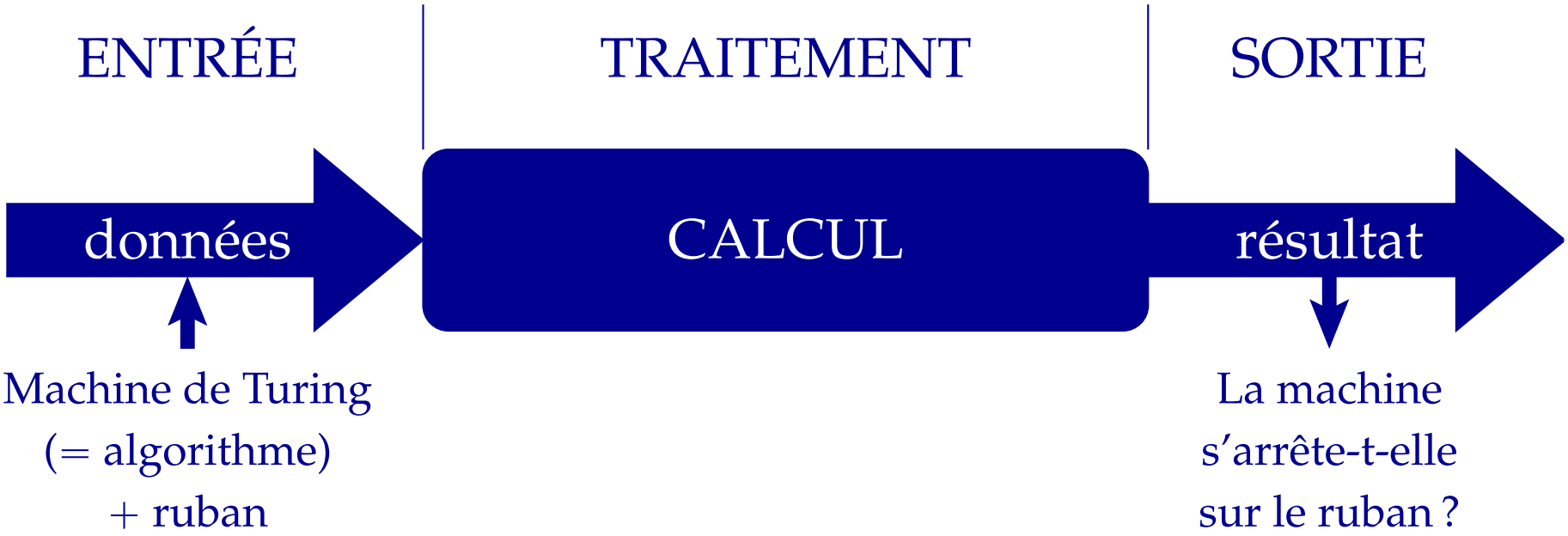
Peut-on tout calculer ?



Pour tout problème, existe-t-il une machine de Turing le résolvant ?

avec des entrées sous forme
de séquences de symboles et
des sorties du type “oui / non”

Le problème de l'arrêt

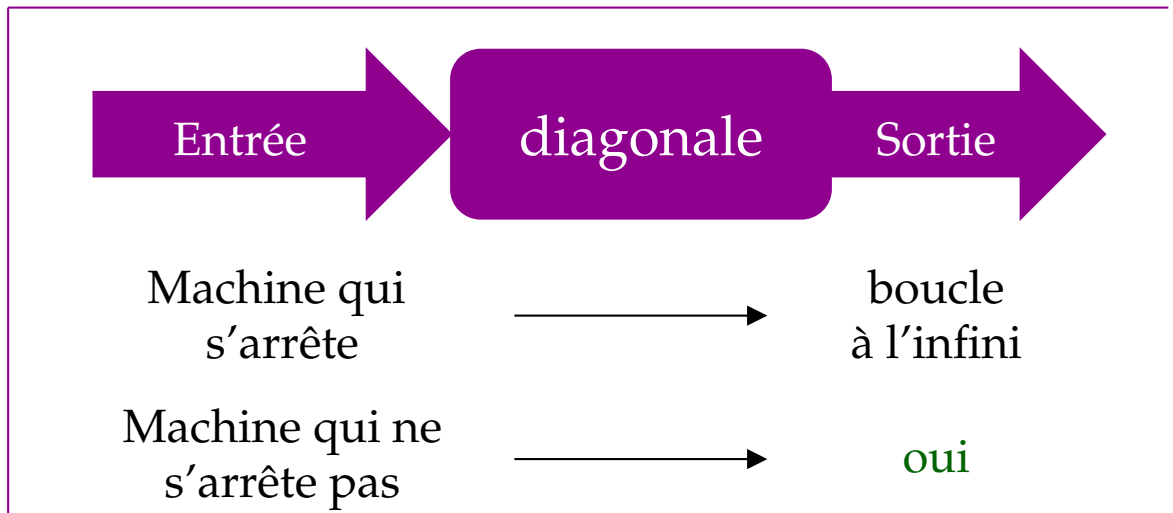


Supposons qu'il existe une machine qui réponde à cette question ?

Le problème de l'arrêt

Construisons la machine réalisant :

Fonction diagonale(M : machine)
  Si (M s'arrête sur le ruban contenant le pseudo-code de M) **alors**
    Tant que (1) faire ;
  Sinon
    retour (oui) ;
  Finsi
Fin



Le problème de l'arrêt

H : existe-t-il un algorithme qui, étant donné n'importe quel algorithme en paramètre, décide s'il s'arrête ?

Démonstration

Par l'absurde.

Supposons qu'un tel algorithme, noté A_H, existe et est tel que :

$$\forall A,\ A_H(A) = \begin{cases} 1 & \text{si } A \text{ s'arrête,} \\ 0 & \text{sinon.} \end{cases}$$

Puisque A_H existe, on peut construire $\overline{A_H}$ tel que :

$$\forall A,\ \overline{A_H}(A) = \begin{cases} \text{boucle } \infty & \text{si } A_H(A) = 1, \\ \text{s'arrête} & \text{sinon.} \end{cases}$$

Or, par définition, l'exécution de $\overline{A_H}(\overline{A_H})$ s'arrête si $\overline{A_H}$ ne s'arrête pas et ne s'arrête pas si $\overline{A_H}$ s'arrête, ce qui est une contradiction. Donc $\overline{A_H}$ n'existe pas, ce qui implique que A_H n'existe pas. □

Plan du cours

La 2nde guerre mondiale

- Nombreuses techniques informatiques développées à cette période, autour :
  - de l'arithmétique (cryptographie, cryptanalyse)
  - des systèmes dynamiques (radars et cibles)
  - des probabilités (méthode de Monte-Carlo)
- Volonté de comprendre les relations mathématiques entre les comportements observés chez les êtres vivants et les machines
  - Analogies entre les couples (radar – cible) et (prédateur – proie)
- Alan Turing et John von Neumann :

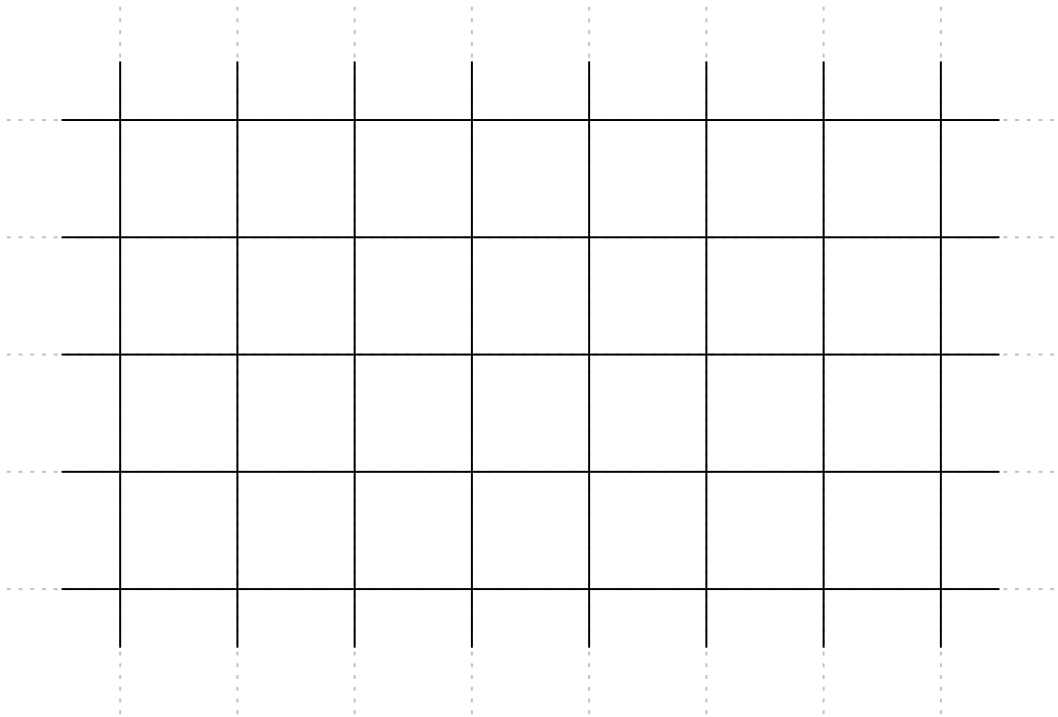
  → Décrire les machines (artificielles) comme des "êtres vivants"
- John von Neumann pose la question :

  → Peut-on construire une machine capable de s'auto-reproduire ?

Formalisation

Idées générales pour créer une telle machine :

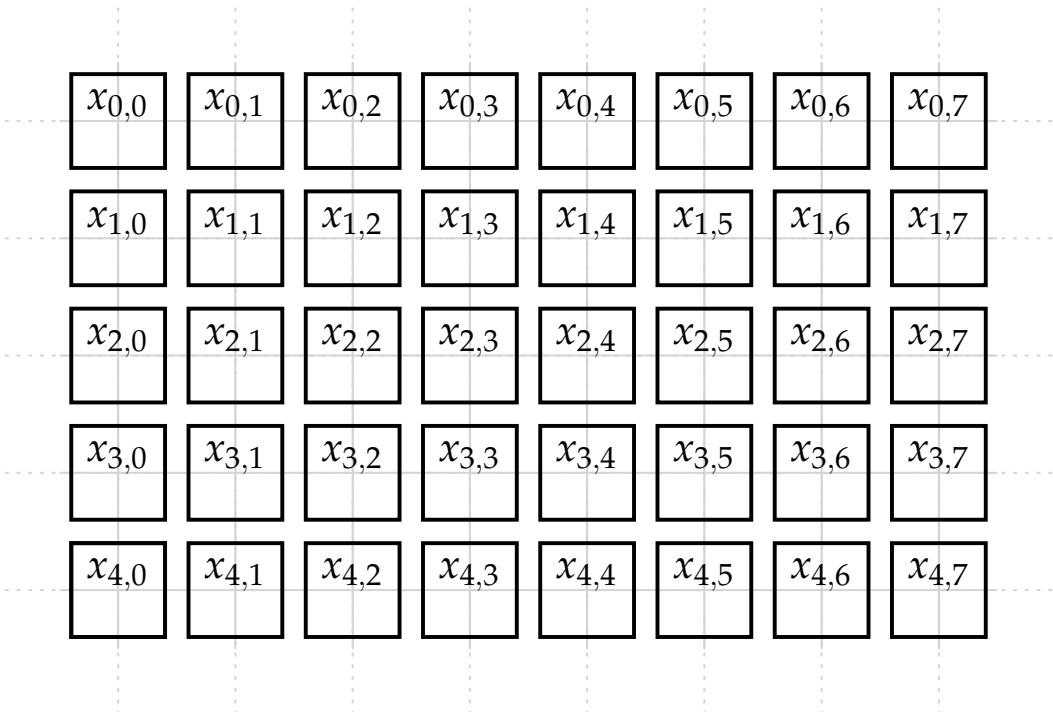
1. Se munir d'un espace mathématique simple qui est une métaphore du monde réel (*e.g.*, $\mathbb{Z}^2$)
2. Placer régulièrement dans cet espace des éléments qui possèdent tous un état qui leur est propre
3. Affecter à l'ensemble des éléments une unique fonction de transition.



Formalisation

Idées générales pour créer une telle machine :

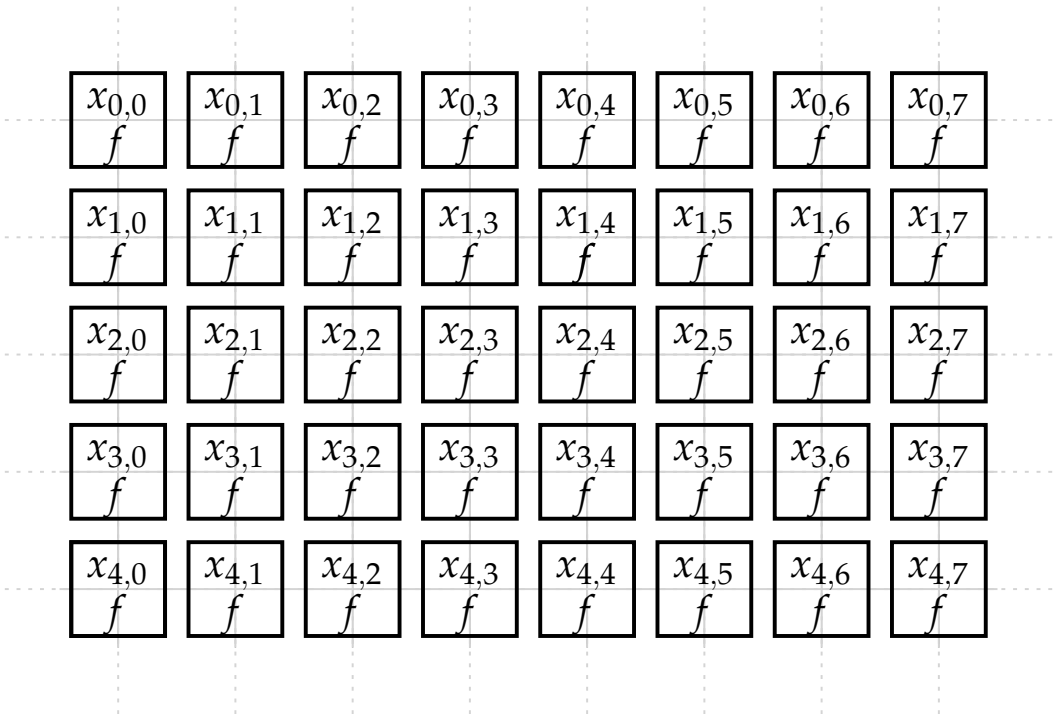
1. Se munir d'un espace mathématique simple qui est une métaphore du monde réel (*e.g.*, $\mathbb{Z}^2$)
2. Placer régulièrement dans cet espace des éléments qui possèdent tous un état qui leur est propre
3. Affecter à l'ensemble des éléments une unique fonction de transition.



Formalisation

Idées générales pour créer une telle machine :

1. Se munir d'un espace mathématique simple qui est une métaphore du monde réel (*e.g.*, $\mathbb{Z}^2$)
2. Placer régulièrement dans cet espace des éléments qui possèdent tous un état qui leur est propre
3. Affecter à l'ensemble des éléments une unique fonction de transition.



Formalisation

Définition

Une cellule est un quadruplet $(d, \mathscr{Q}, V, f)$ où :

- d est la dimension
- $\mathscr{Q}$ est un ensemble fini d'états, *i.e.*, son alphabet
- $V \subseteq \mathbb{Z}^d$ est son voisinage
- $f : \mathscr{Q}^a \to \mathscr{Q}$ est sa fonction de transition, avec $a = |V|$ son arité

Définition

Un automate cellulaire $\mathscr{A} \subseteq \mathbb{Z}^d$ est un sous-ensemble de l'espace discret de dimension d dont chaque point de $\mathscr{A}$ est une cellule de fonction f.

Formalisation

Exemple

c_k

$$d = 1$$
$$\mathcal{Q} = \{0,1\}$$
$$V = \{c_{k-1}, c_k, c_{k+1}\}$$
$$f(c_k) = c_{k-1} +_2 c_k +_2 c_{k+1}$$

ACE 150

Diagramme espace-temps ($t\uparrow$)

Récapitulatif

- Localité :
  - ne dépend que du voisinage
- Homogénéité (invariance par translation) :
  - toutes les cellules évoluent en suivant la même règle et ont le même voisinage
- Parallélisme massif :
  - à chaque étape de temps, toutes les cellules mettent à jour leur état
- Manières de voir l'espace (cellulaire) :
  - infini
  - fini borné
  - fini périodique (tore de dimension d)

Machines de Turing vs automates cellulaires

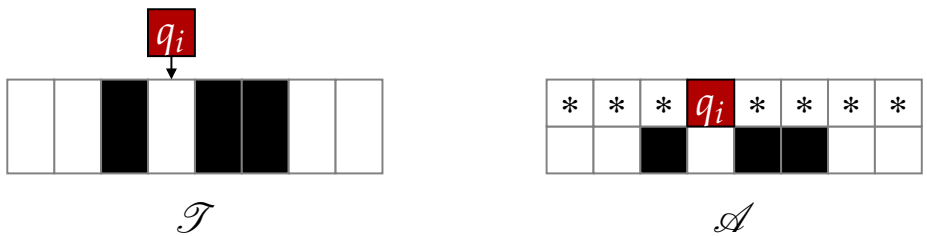
Théorème (Smith, 1971)

Toute machine de Turing $\mathscr{T} = (Q_{\mathscr{T}}, \Sigma, \delta)$ peut être simulée par un automate cellulaire $\mathscr{A} = (\mathbb{Z}, V_0 = (-1,0,1), Q_{\mathscr{A}}, f)$.

Idée générale

$\mathscr{A}$ est capable de simuler $\mathscr{T}$ si :

- l'alphabet de $\mathscr{A}$ est $\Sigma \times (Q \cup \{*\})$: l'état d'une cellule est (σ, q) où σ est le contenu du ruban et q indique la présence ou l'absence de la tête de $\mathscr{T}$ et l'état de $\mathscr{T}$ le cas échéant
- son voisinage est $\{-1,0,1\}$
- $\mathscr{A}$ reproduit les mouvements de la tête de $\mathscr{T}$ et les changements d'états avec q et les modifications de ruban avec σ



$\mathscr{T}$ $\mathscr{A}$

Plan du cours

Définition

- États : $\mathscr{Q} = \{0,1\}$
- Voisinage : $V = \{-1,0,1\}$
- Configuration au temps t :
  $x(t)$: vecteur qui associe un état de $\mathscr{Q}$ à chaque cellule

$$x(t) = (0,1,1,0,0,0,1,0,1,1)$$

- Question : combien existe-t-il de tels automates cellulaires ?

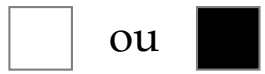
Définition

- États : $\mathscr{Q} = \{0,1\}$
- Voisinage : $V = \{-1,0,1\}$
- Configuration au temps t :
  $x(t)$: vecteur qui associe un état de $\mathscr{Q}$ à chaque cellule

$$x(t) = (0,1,1,0,0,0,1,0,1,1)$$

- Question : combien existe-t-il de tels automates cellulaires ?
  - Combien de configurations de voisinage possibles ? $2^3 = 8$

  - Combien de nouveaux états possibles ? 2



↓

$$2^{2^3} = 256 \text{ automates}$$

Identification de Wolfram

- Objectif : donner un code unique à chaque automate cellulaire élémentaire
- Idée de Stephen Wolfram → leur associer un nombre entre 0 et 255, à savoir une somme de puissances de 2

Identification de Wolfram

- Objectif : donner un code unique à chaque automate cellulaire élémentaire
- Idée de Stephen Wolfram → leur associer un nombre entre 0 et 255, à savoir une somme de puissances de 2

1. Chaque configuration de voisinage correspond à un nombre binaire entre 0 et $2^3 - 1 = 7$

Identification de Wolfram

- Objectif : donner un code unique à chaque automate cellulaire élémentaire
- Idée de Stephen Wolfram → leur associer un nombre entre 0 et 255, à savoir une somme de puissances de 2

1. Chaque configuration de voisinage correspond à un nombre binaire entre 0 et $2^3 - 1 = 7$
2. On associe “trivialement” à chacune une puissance de 2

Identification de Wolfram

- Objectif : donner un code unique à chaque automate cellulaire élémentaire
- Idée de Stephen Wolfram → leur associer un nombre entre 0 et 255, à savoir une somme de puissances de 2

1. Chaque configuration de voisinage correspond à un nombre binaire entre 0 et $2^3 - 1 = 7$
2. On associe “trivialement” à chacune une puissance de 2
3. On affecte un coefficient 0 ou 1 à chaque puissance de 2 en fonction du nouvel état obtenu après application de la règle

0×2^0 1×2^1 1×2^2 0×2^3 1×2^4 0×2^5 1×2^6 0×2^7

Identification de Wolfram

- Objectif : donner un code unique à chaque automate cellulaire élémentaire
- Idée de Stephen Wolfram → leur associer un nombre entre 0 et 255, à savoir une somme de puissances de 2

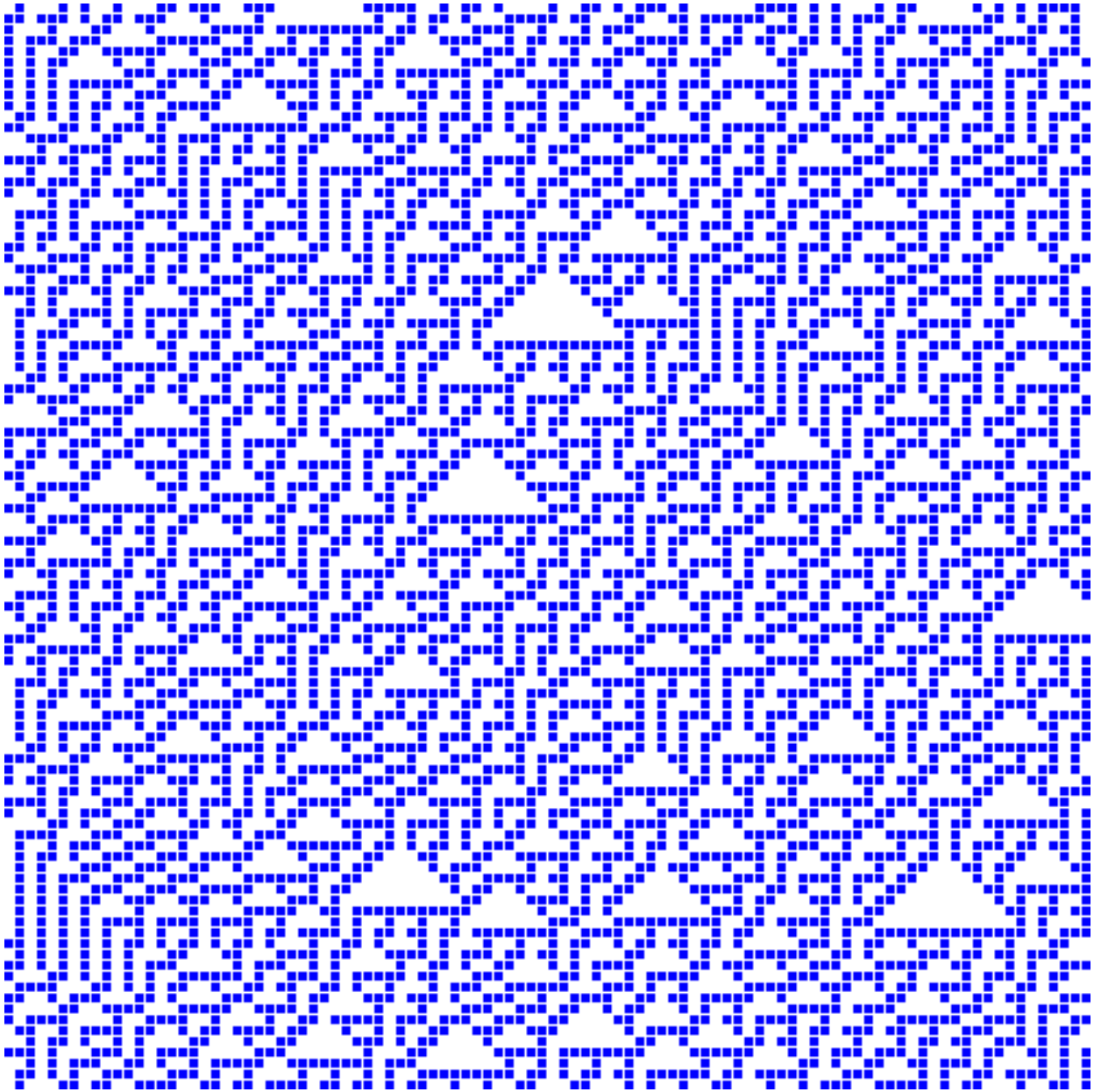
1. Chaque configuration de voisinage correspond à un nombre binaire entre 0 et $2^3 - 1 = 7$
2. On associe "trivialement" à chacune une puissance de 2
3. On affecte un coefficient 0 ou 1 à chaque puissance de 2 en fonction du nouvel état obtenu après application de la règle
4. On fait la somme de tout ça

$0\times 2^0 + 1\times 2^1 + 1\times 2^2 + 0\times 2^3 + 1\times 2^4 + 0\times 2^5 + 1\times 2^6 + 0\times 2^7$

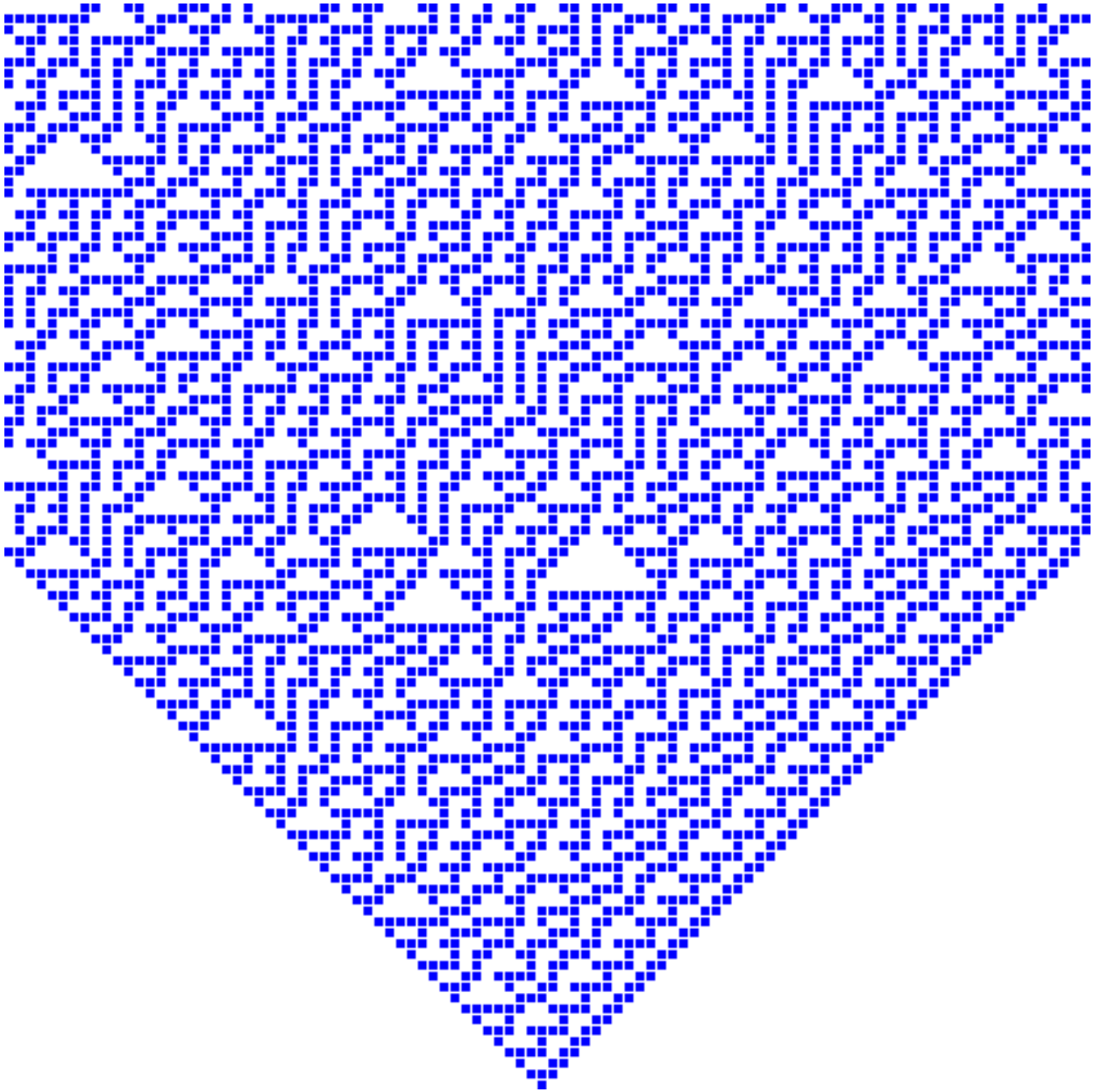
D'où la règle $2+4+16+64 = 86$

Évolution de la règle 86

Évolution de la règle 86



Évolution de la règle 86



Classification

- Stephen Wolfram, 1984 :
  - Simulations numériques “large échelle” des automates cellulaires élémentaires
  - Mise en évidence que certains automates ont des caractéristiques comportementales proches
  - Attention aux biais de simulation lié à un échantillonnage peu rigoureux

→ Classification selon leur “complexité” comportementale

I	Uniformité	L’automate atteint une configuration homogène
II	Périodicité	L’automate aboutit à une phase de répétition périodique de configuration.s
III	Chaos	L’automate mène à des motifs chaotiques non périodiques
IV	Complexité	L’automate fait apparaître des particules “animées d’une existence propre”

Classe des automates uniformes

ACE 8

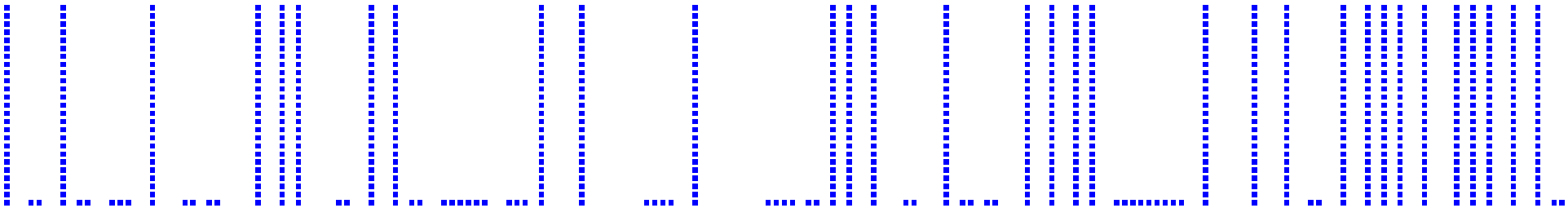
ACE 32

Classe des automates uniformes

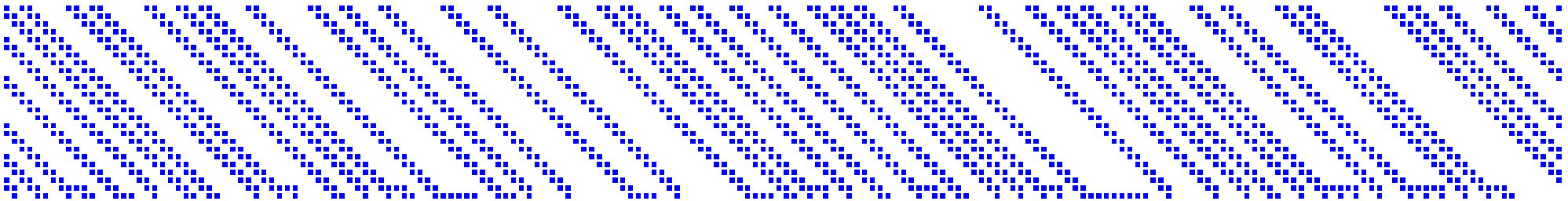
Simuler en live l'ACE 128

Classe des automates périodiques

ACE 4



ACE 46

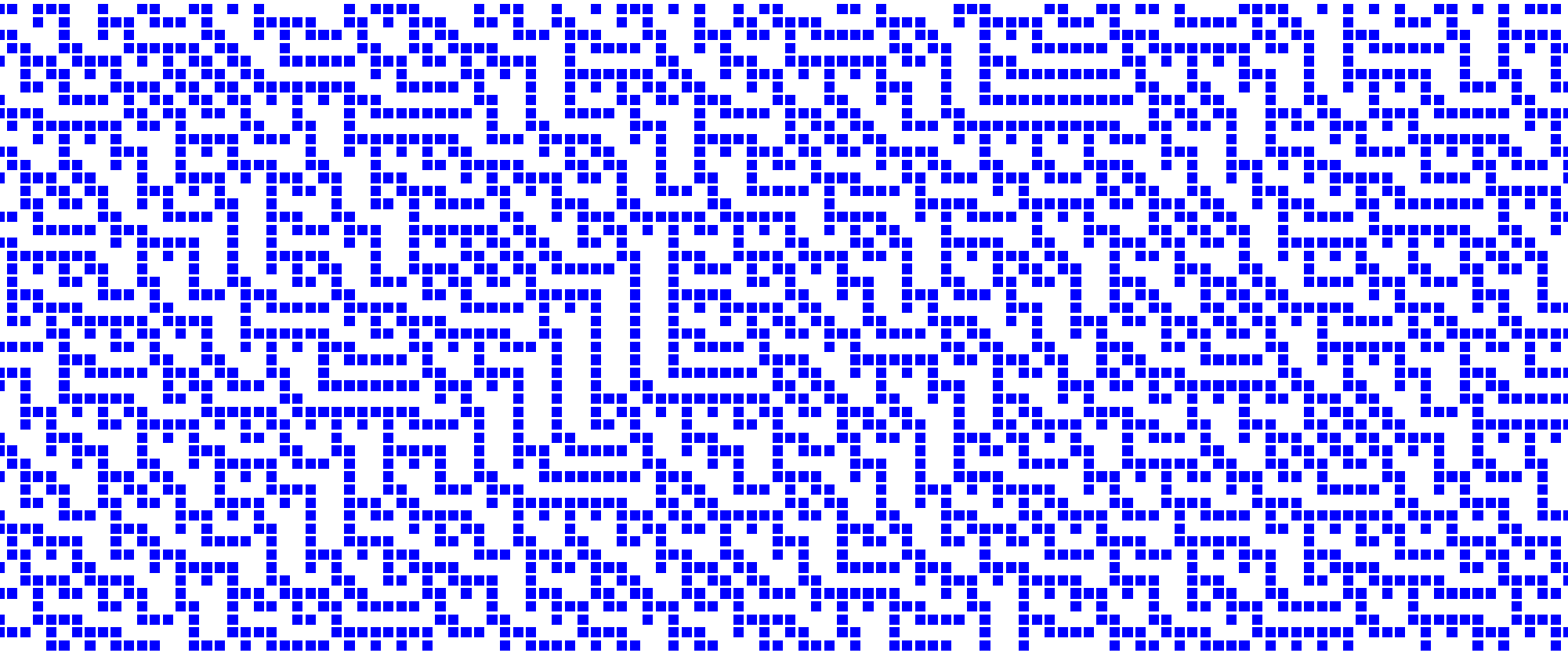


Classe des automates périodiques

Simuler en live l'ACE 57 avec différentes densités

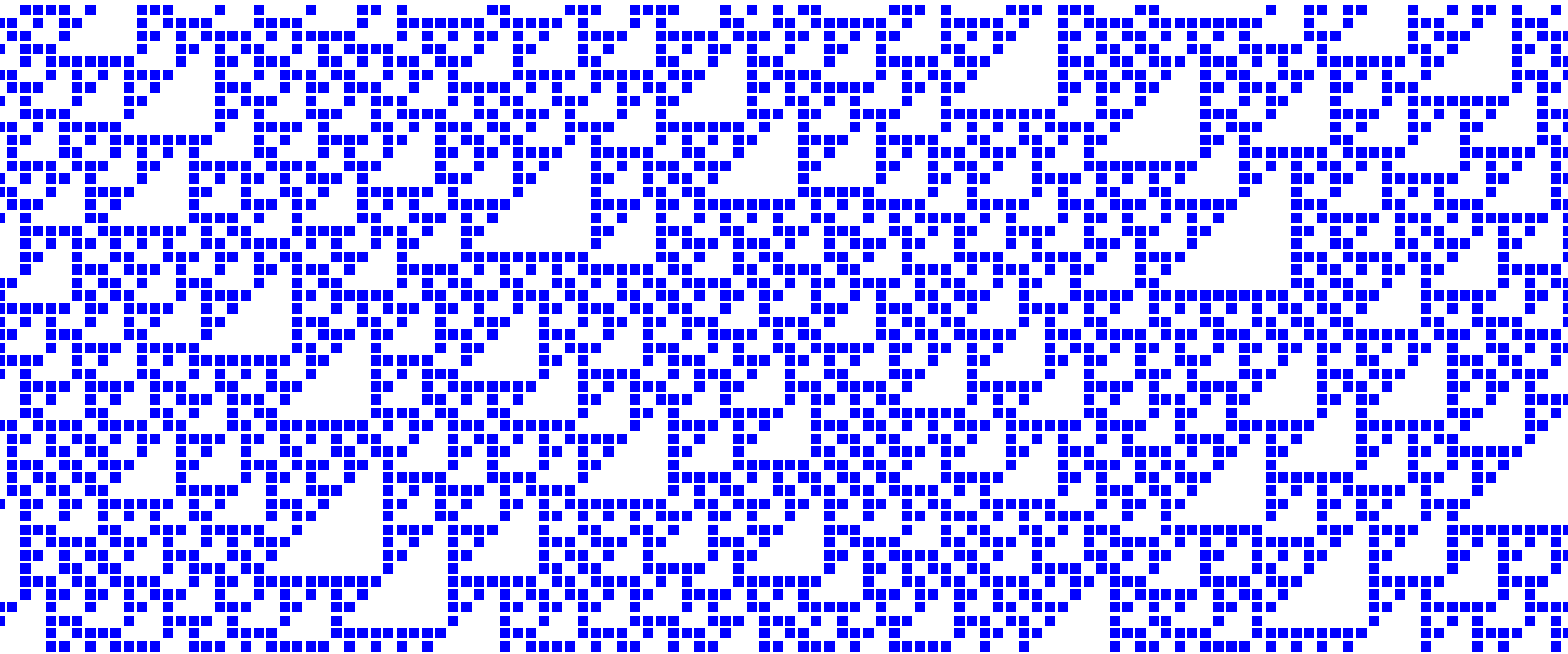
Classe des automates chaotiques

ACE 45



Classe des automates chaotiques

ACE 60



Classe des automates chaotiques

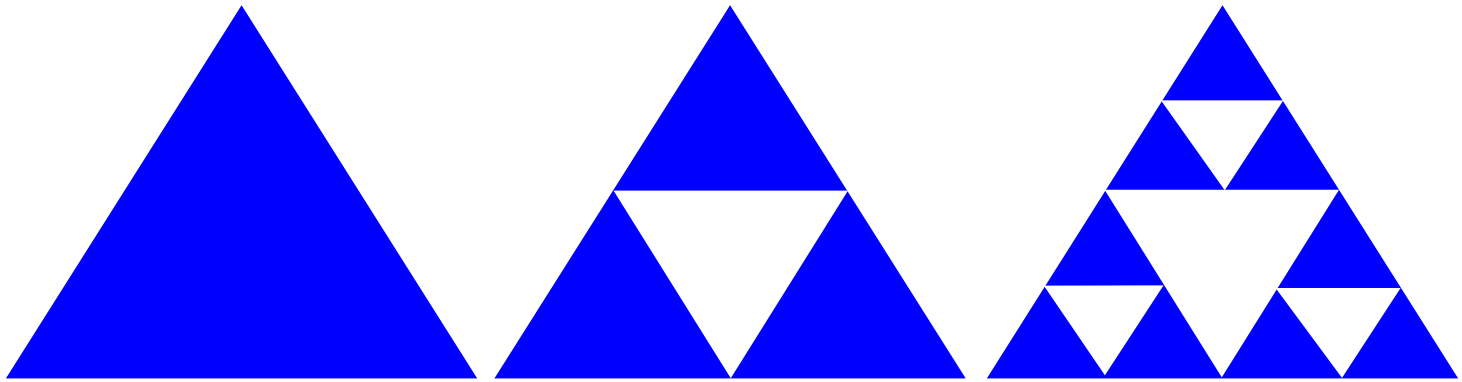
Regardons ça de plus près

- “L’automate mène à des motifs chaotiques non périodiques”
- Ressemble à des comportements aléatoires
- On peut structurer mathématiquement l’espace des configurations et décrire certaines propriétés de leur évolution
  → à tel point qu’on peut avoir des comportements fractals
- En résumé, c’est compliqué

Classe des automates chaotiques

Triangle de Sierpiński

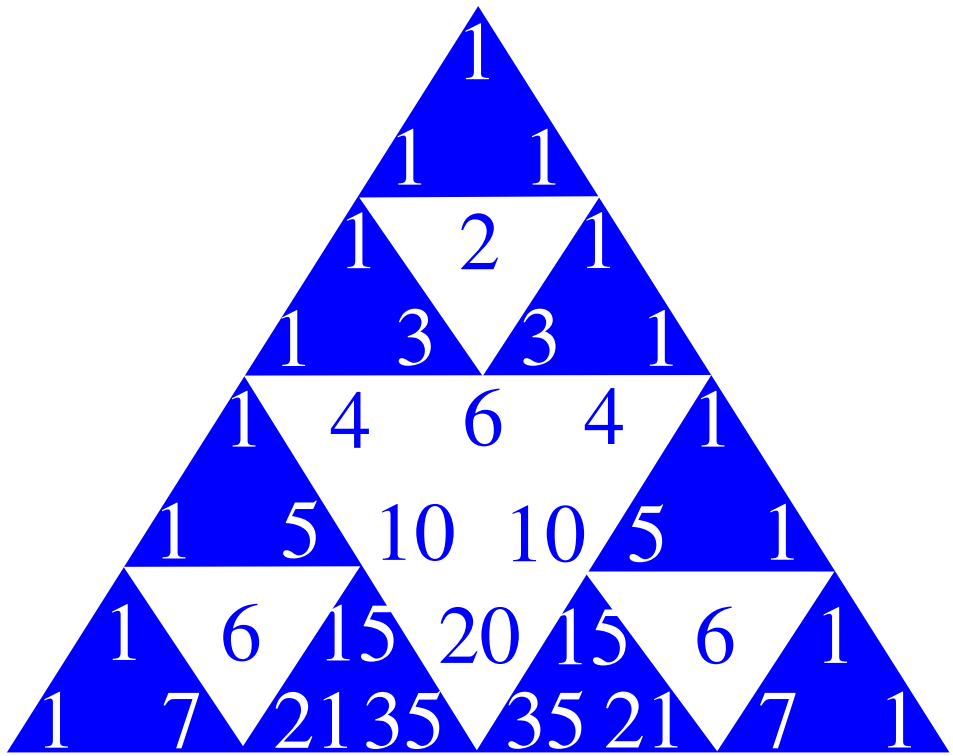
- Méthode n°1 : évider itérativement un triangle équilatéral



Classe des automates chaotiques

Triangle de Sierpiński

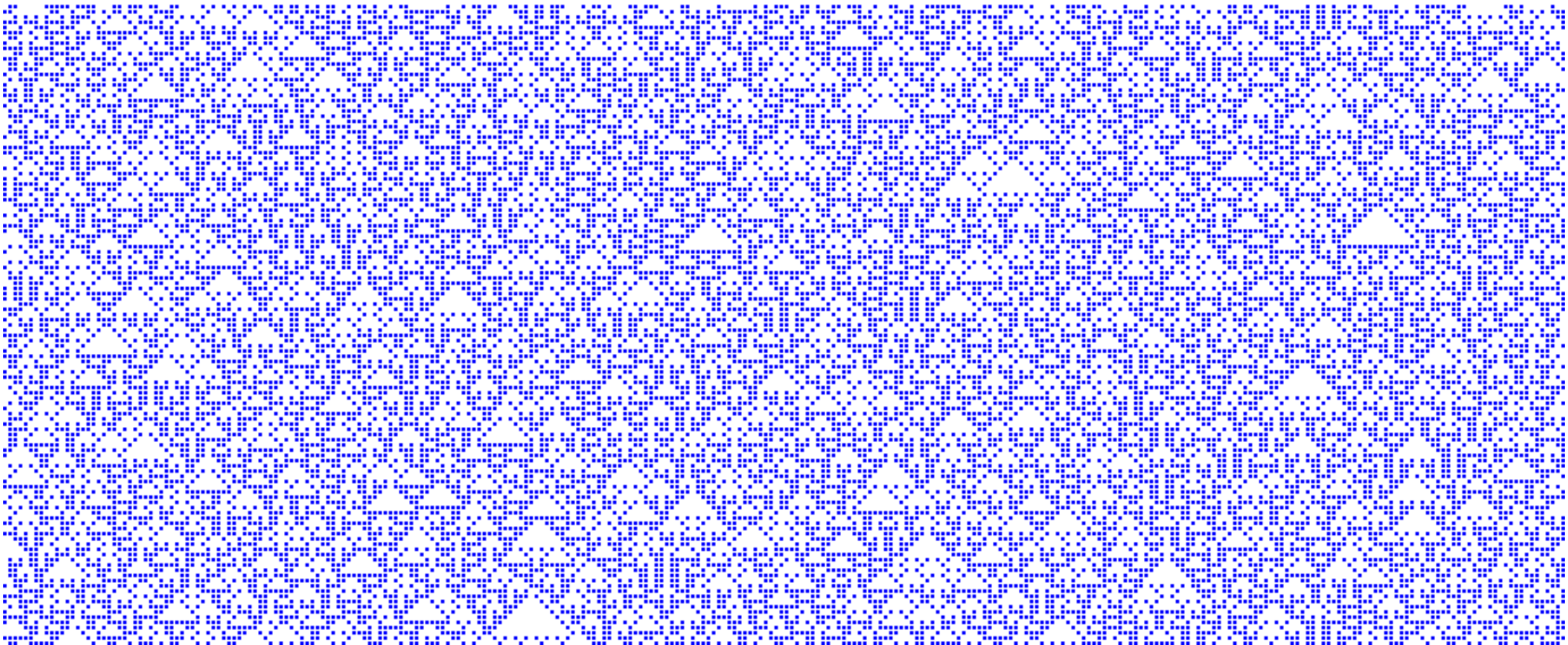
- Méthode n°1 : évider itérativement un triangle équilatéral
- Méthode n°2 : dessiner le triangle de Pascal



Classe des automates chaotiques

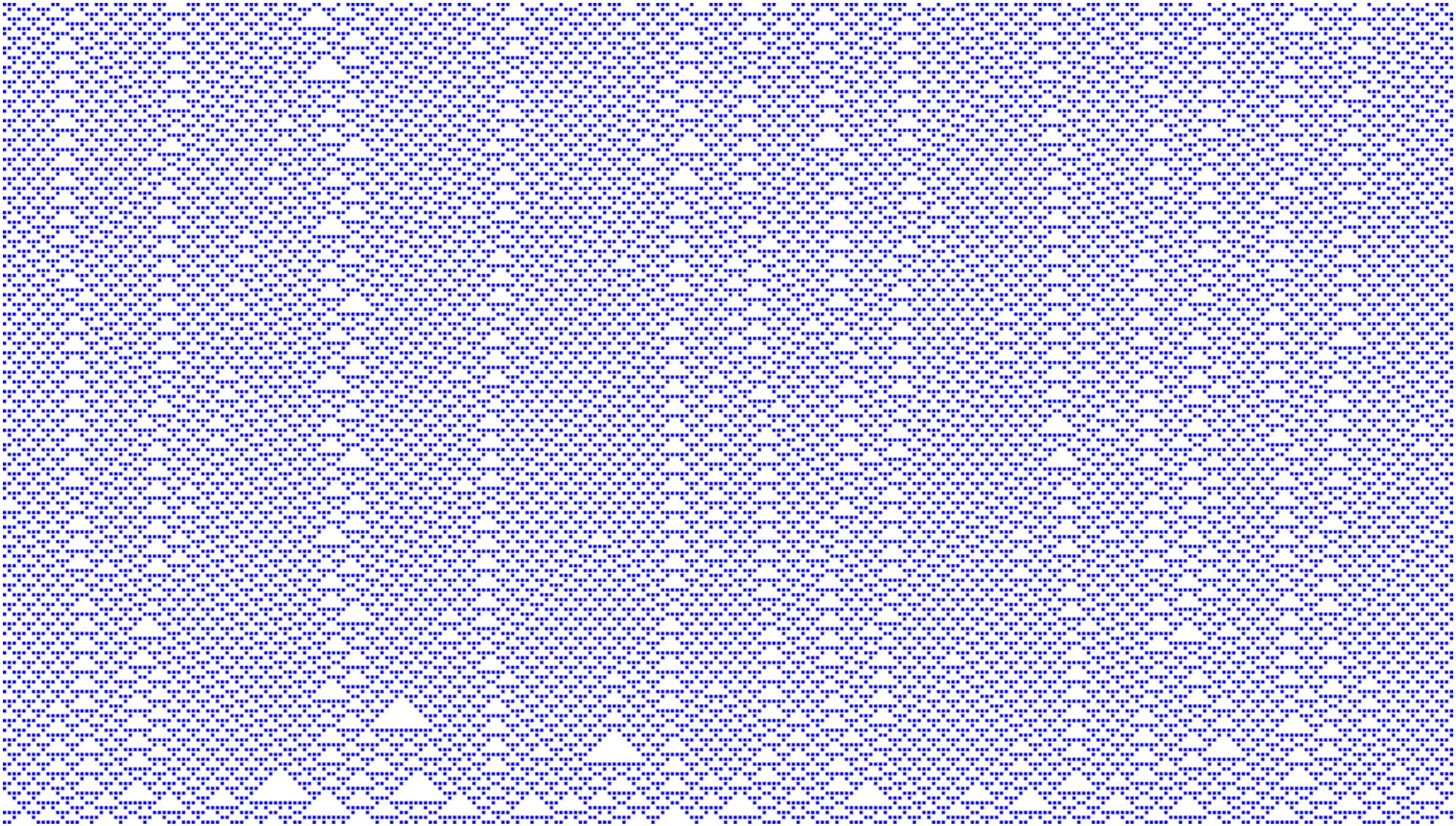
Triangle de Sierpiński

- Méthode n°1 : évider itérativement un triangle équilatéral
- Méthode n°2 : dessiner le triangle de Pascal
- Méthode n°3 : simuler (en live) la règle 90 que voici (avec densité 0,5)



Classe des automates complexes

ACE 54



Classe des automates complexes

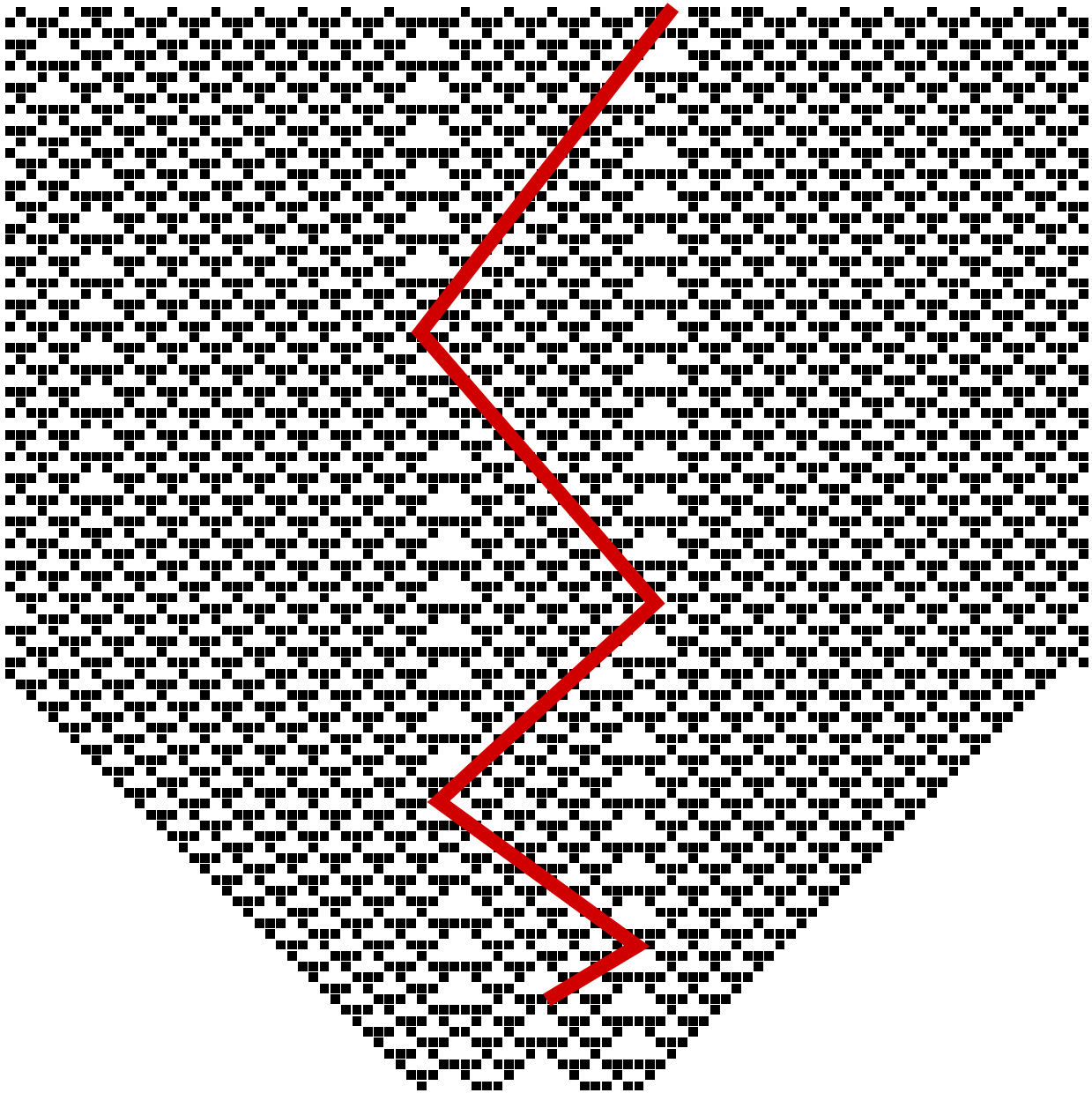
Regardons ça de plus près

- “L’automate fait apparaître des particules animées d’une existence propre”
- Présence de “particules” vraiment spéciales
- Étonnante forme d’ordre local surgissant de façon imprévisible qui est plus que compliquée et reste non décrite mathématiquement dans le cas général
- En résumé, c’est complexe

Classe des automates complexes

Simuler en live l'ACE 110

Parabole discrète



Firing squad

- Problème proposé par John Myhill en 1957 : construire un AC qui, en partant d'une unique cellule active, finit dans une configuration où toutes les cellules sont actives simultanément.

- Largement étudié par la suite :

  - 1ère solution (non optimale en temps) donnée par John McCarthy et Marvin Minsky, 1962

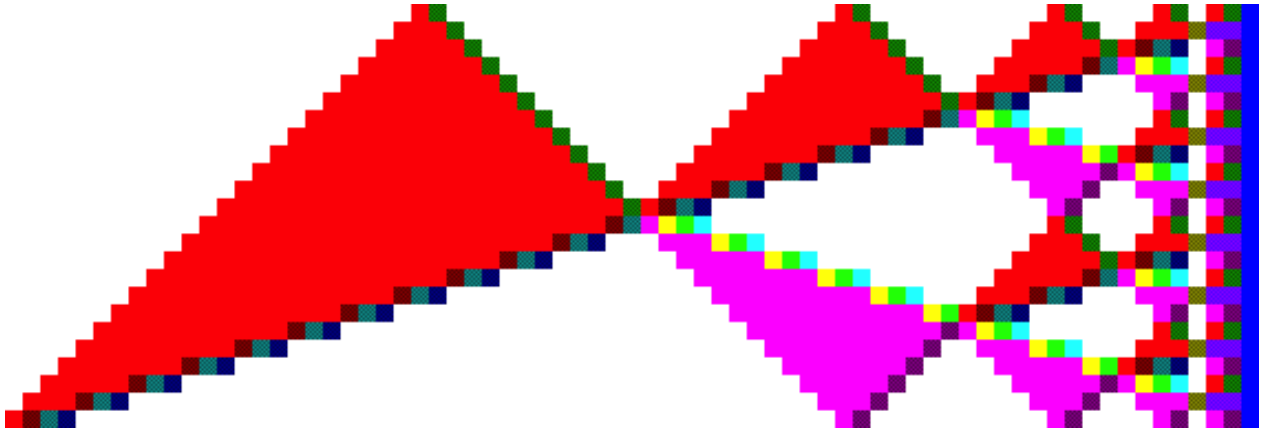
  - Solutions optimales ($2n-2$ étapes de temps) :
    - ◇ Eiichi Goto, 1962 (plus de 1000 états)
    - ◇ Abraham Waksman, 1966 (16 états)
    - ◇ Robert Balzer, 1967 (8 états)
    - ◇ Jacques Mazoyer, 1986 (7 états), 1987 (6 états)

Firing squad

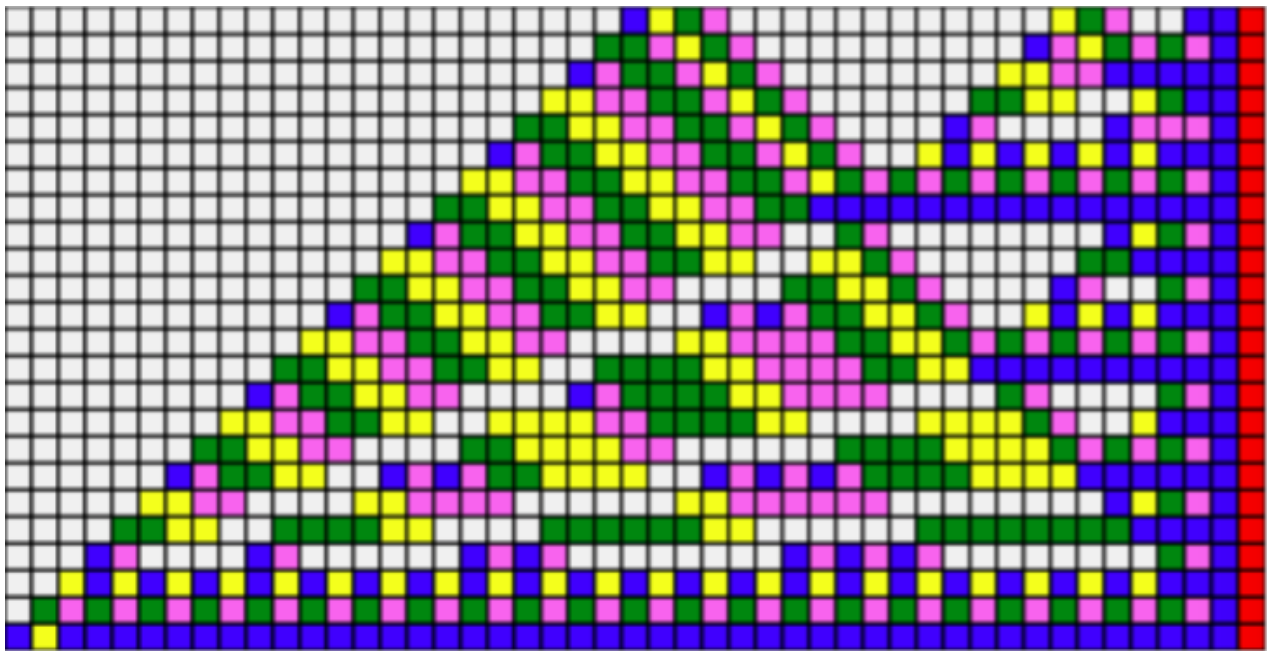
Nombre d'états	Temps optimal	Temps non optimal
3	Pas de solution (Balzer, 1967)	Pas de solution (Yunes, 1993)
4	Pas de solution (Sanders, 1994)	Question ouverte
5	Question ouverte	Question ouverte
6	Oui (Mazoyer, 1987)	—
7	Oui (Mazoyer, 1986)	—
8	Oui (Balzer, 1967)	—

Firing squad

- Solution non optimale à 15 états



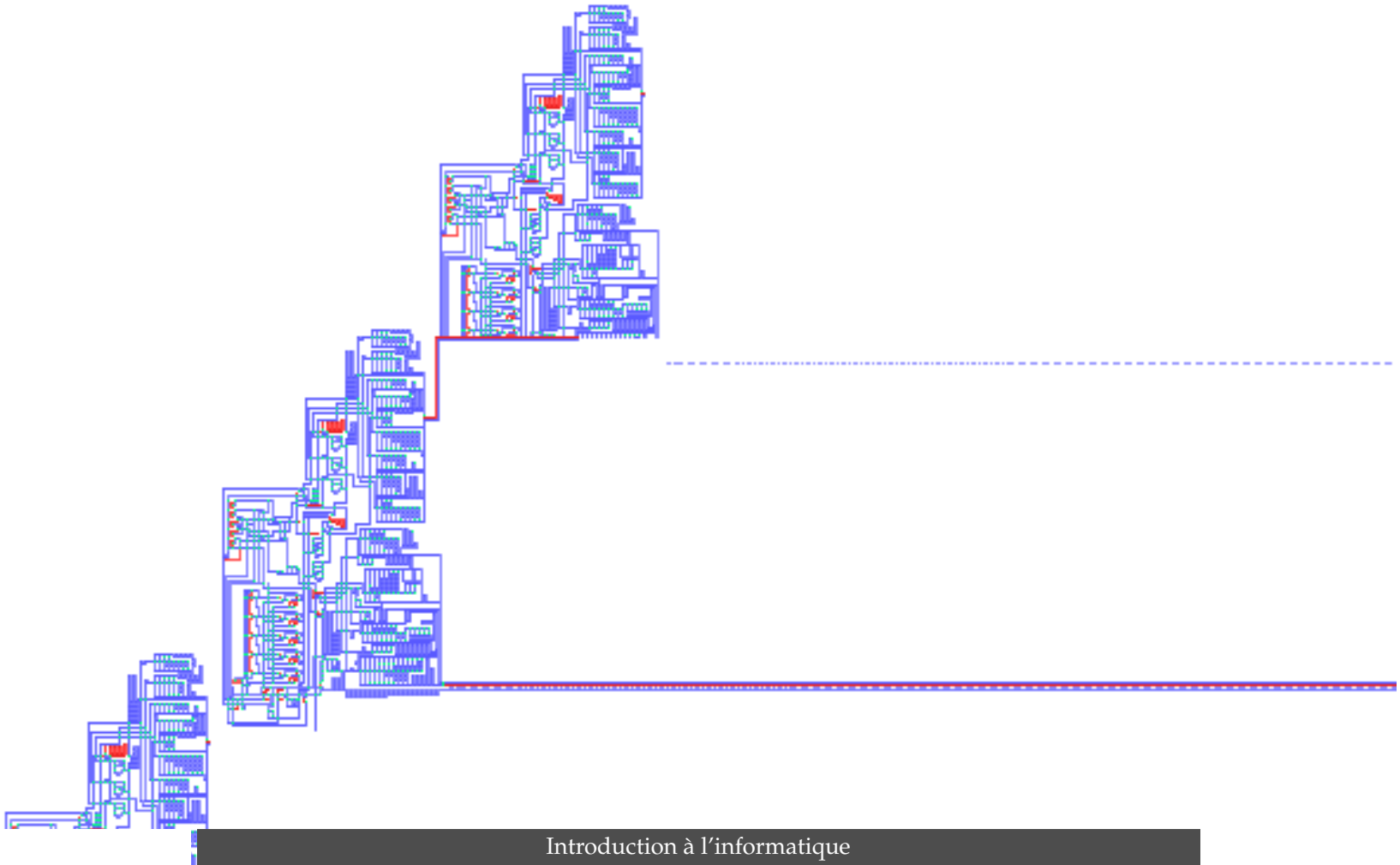
- Solution optimale à 6 états de Mazoyer



Plan du cours

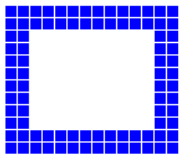
Auto-réplication

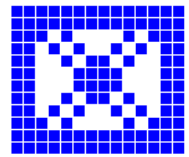
- von Neumann y est parvenu
- Conception du constructeur universel
- Automate cellulaire à 29 états



Auto-réplication

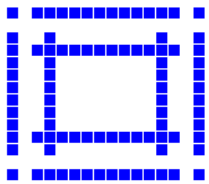
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

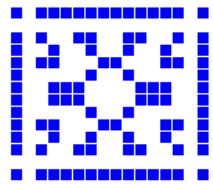




Auto-réplication

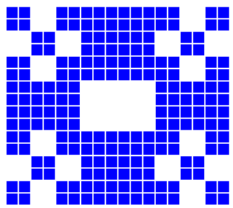
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

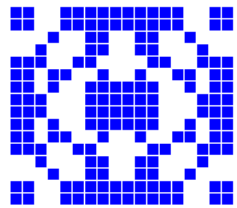




Auto-réplication

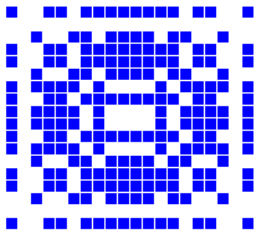
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

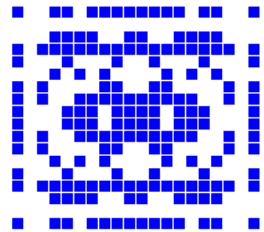




Auto-réplication

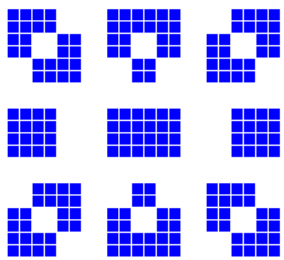
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

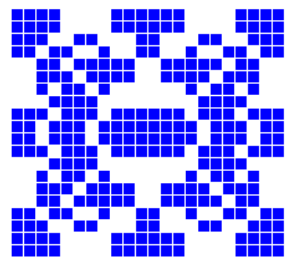




Auto-réplication

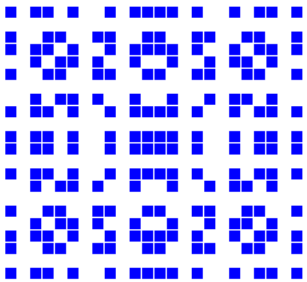
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

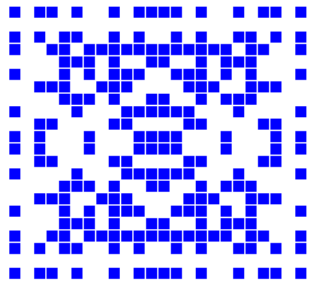




Auto-réplication

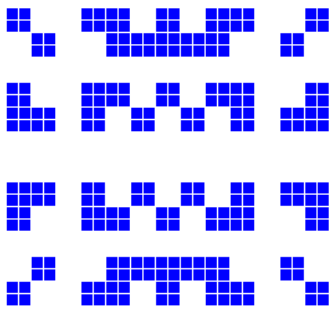
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

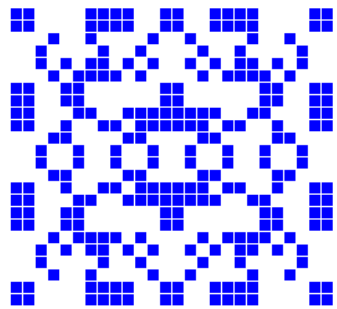




Auto-réplication

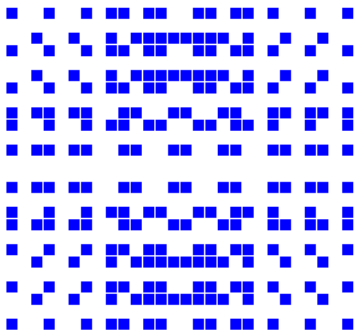
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

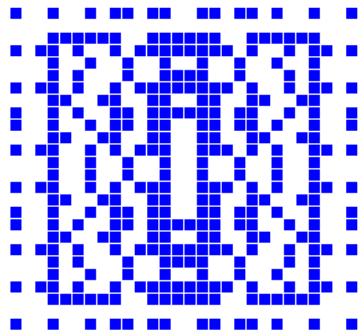




Auto-réplication

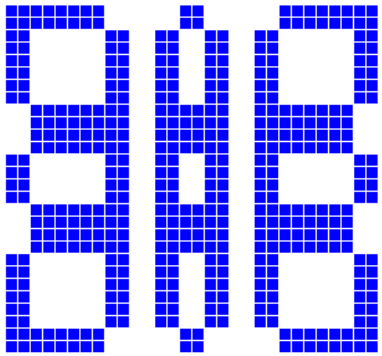
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

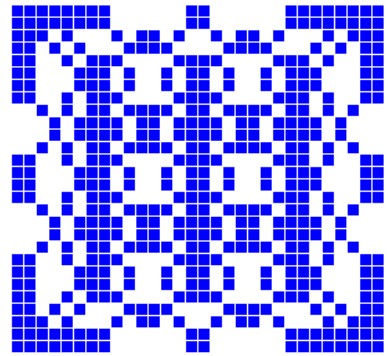




Auto-réplication

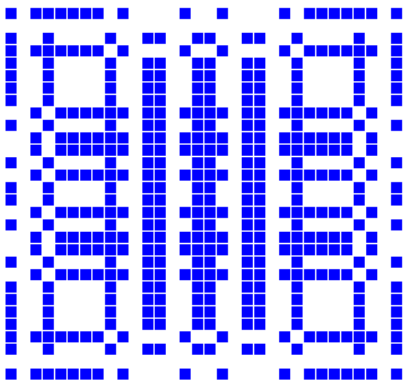
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

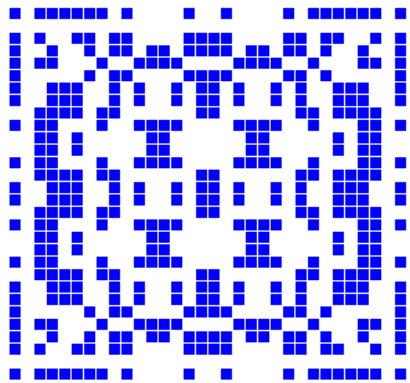




Auto-réplication

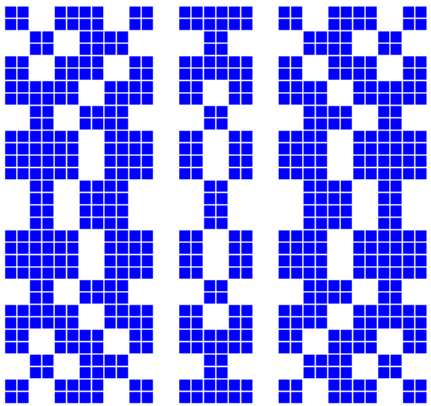
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

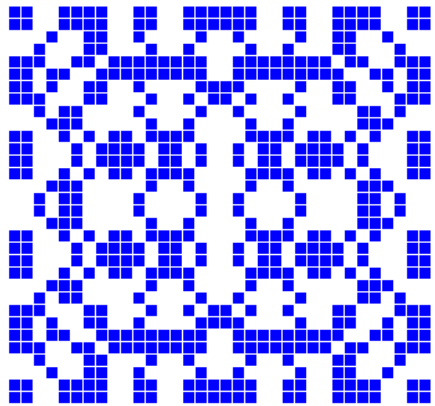




Auto-réplication

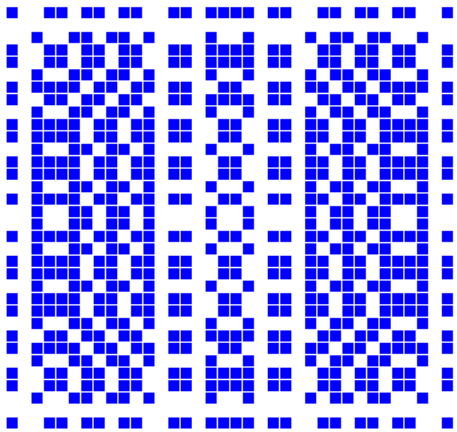
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

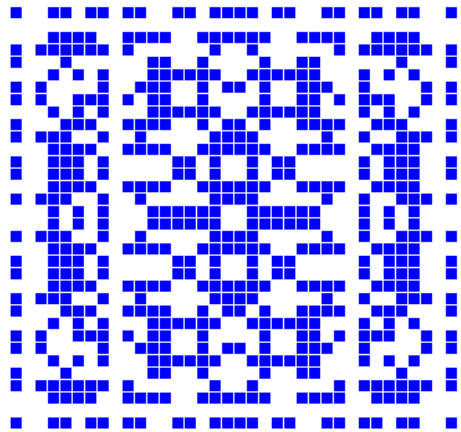




Auto-réplication

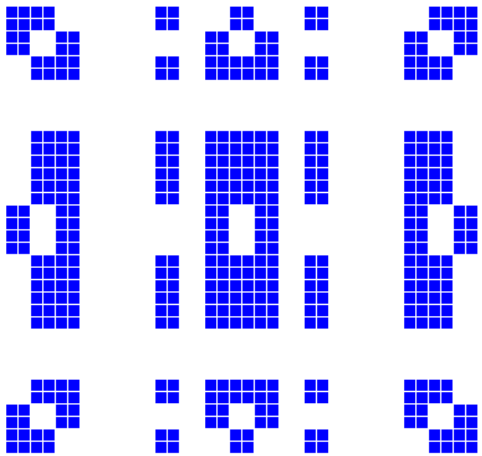
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

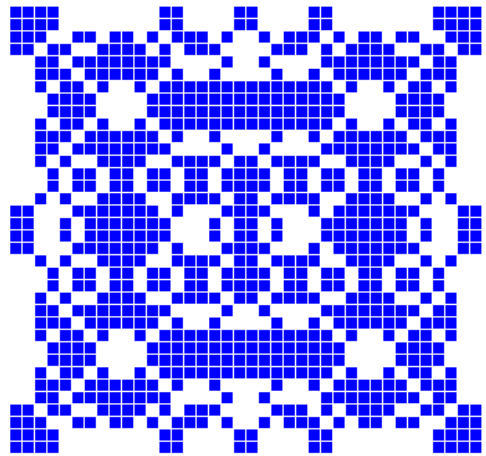




Auto-réplication

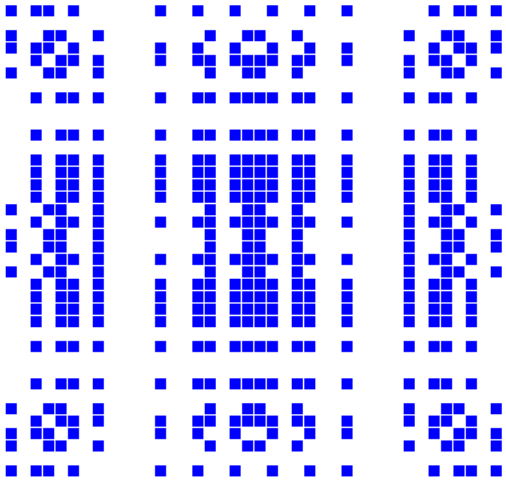
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

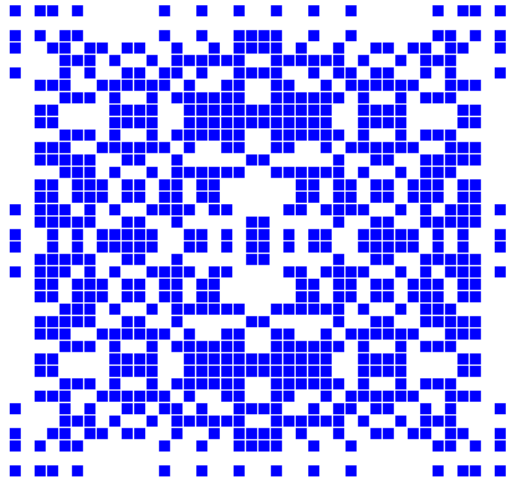




Auto-réplication

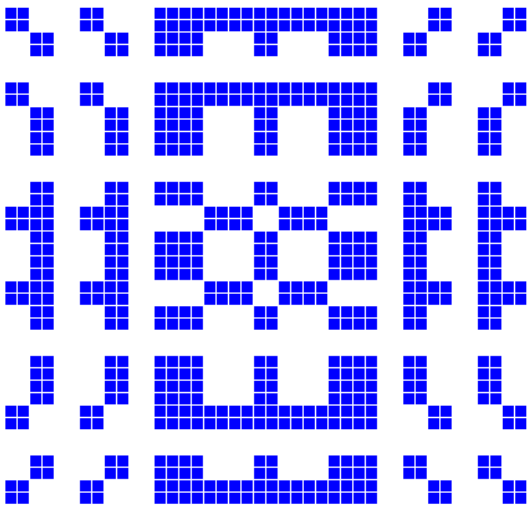
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

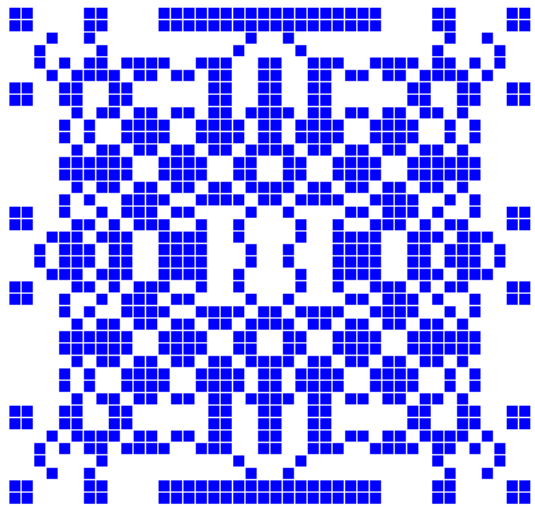




Auto-réplication

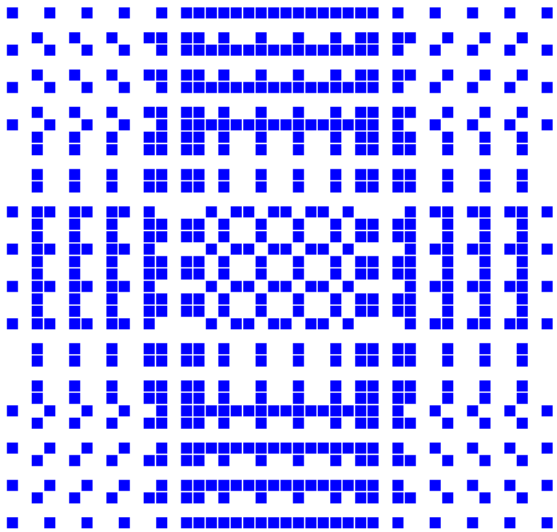
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

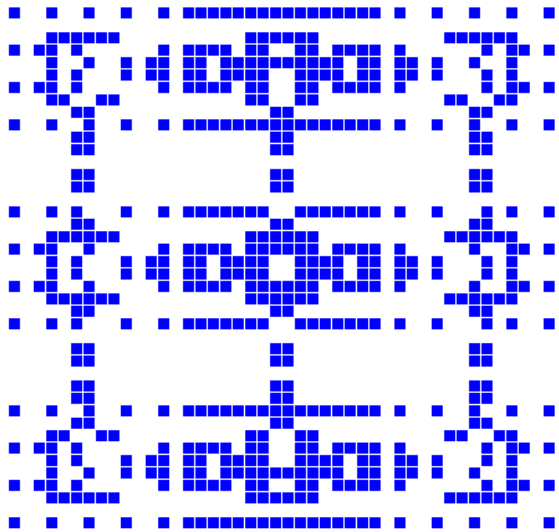




Auto-réplication

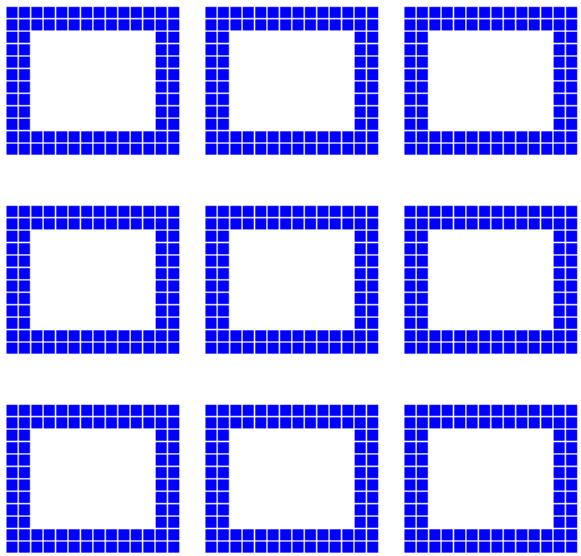
- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial

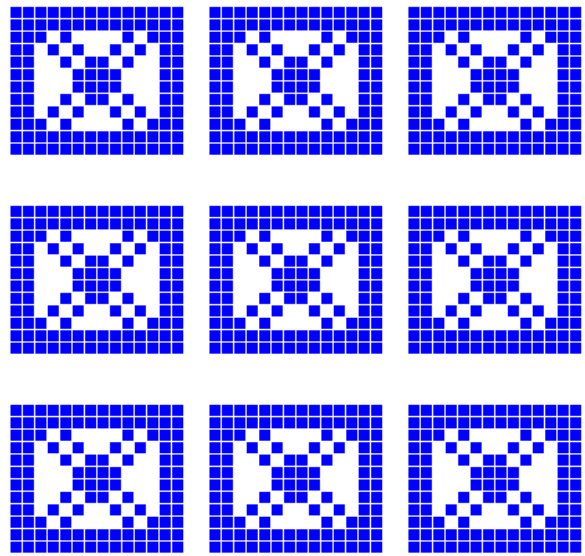




Auto-réplication

- Mais c'est "compliqué" comme réplication...
- ... et Edward Friedkin a montré qu'il y avait plus simple (bien que non universel) avec le compteur de parité (règle XOR, voisinage de Moore)
- Théorème : les réplicats sont distants de la puissance de 2 immédiatement supérieure à la plus grande des longueurs du motif initial





Majorité et minorité

- Automate majorité
  - Voisinage de Moore ou de von Neumann
  - À chaque étape de temps, chaque cellule prend l'état majoritaire dans son voisinage
- Automate minorité
  - Voisinage de Moore ou de von Neumann
  - À chaque étape de temps, chaque cellule prend l'état minoritaire dans son voisinage

- Attention à l'intuition pouvant amener à penser que les comportements de ces 2 automates sont symétriques

- Simulation en live

Jeu de la vie

Bases

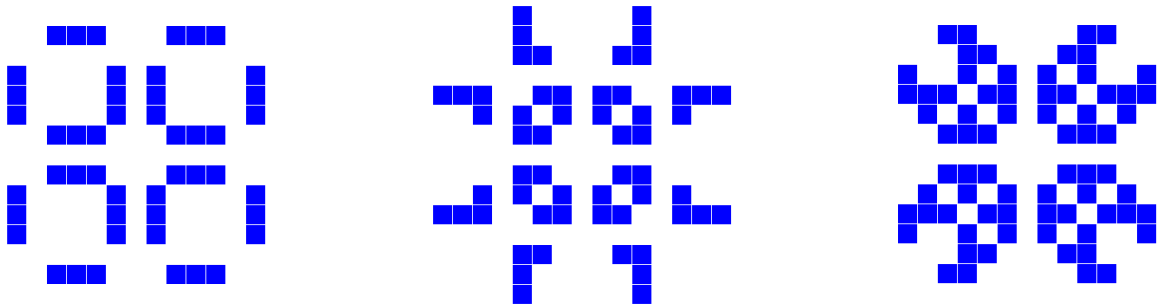
- Inventé par John Conway en 1970
- Automate à 2 états (vivant ou mort) à voisinage de Moore à huit voisins
- Les règles sont (très) simples :
  - Une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît)
  - Une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt
- Simuler le jeu de la vie sur une grille 100×100

Jeu de la vie

Stabilités et périodicités

- Motifs stables

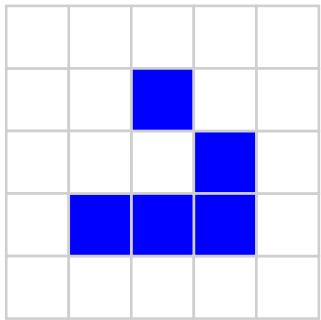
- Motifs périodiques



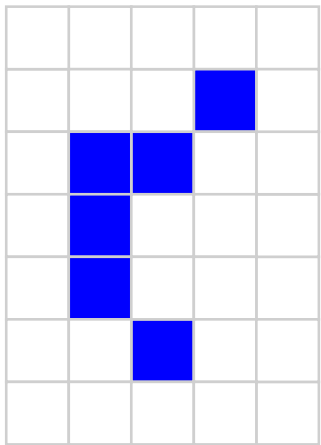
Jeu de la vie

Motifs remarquables

- *Gliders* : motifs oscillants se déplaçant



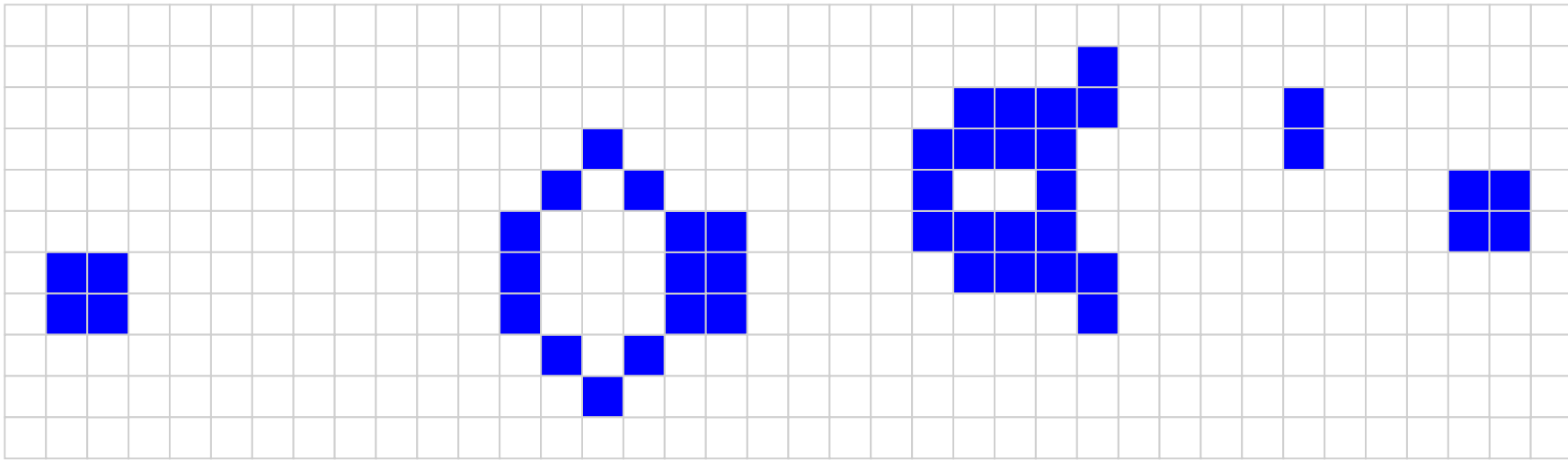
- *Puffers* : motifs oscillants laissant des “débris” derrière eux (voir aussi `https://www.youtube.com/watch?v=oHgLOOJ0mnY`)



Jeu de la vie

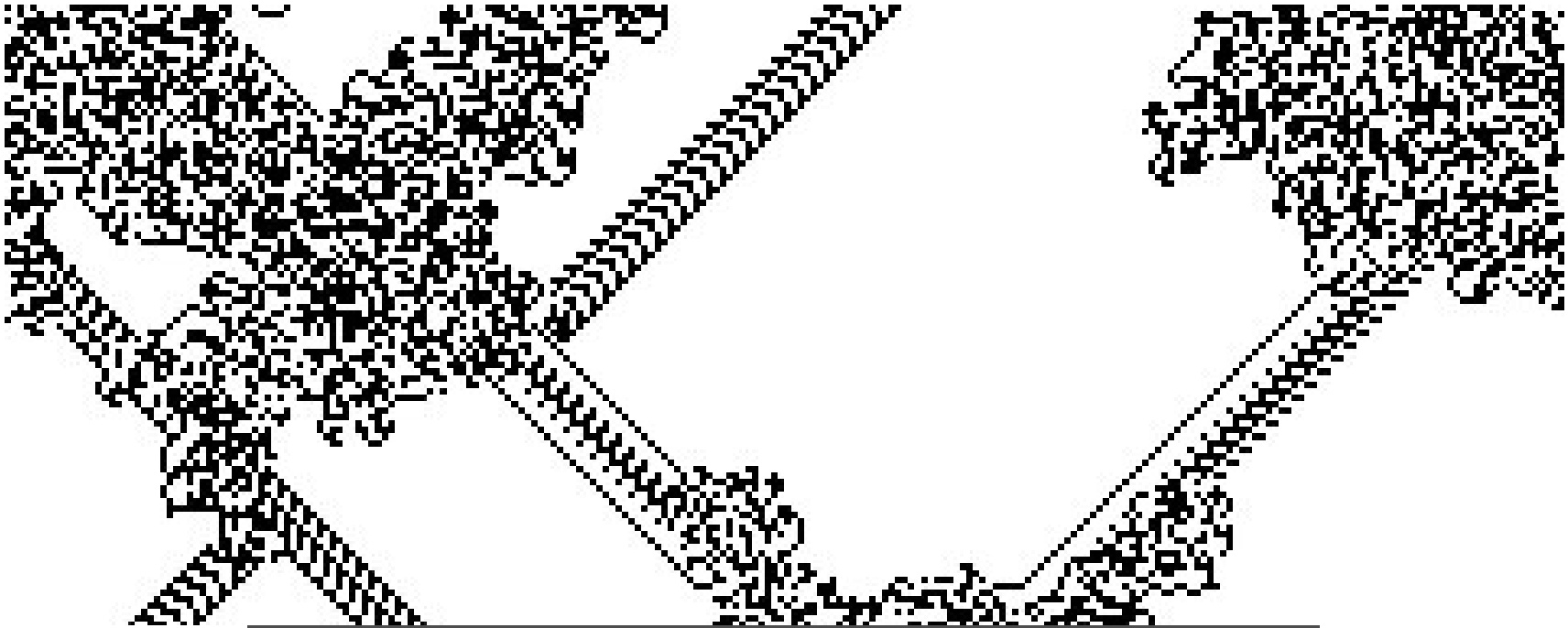
Motifs remarquables

- *Guns* : générateurs de *gliders*



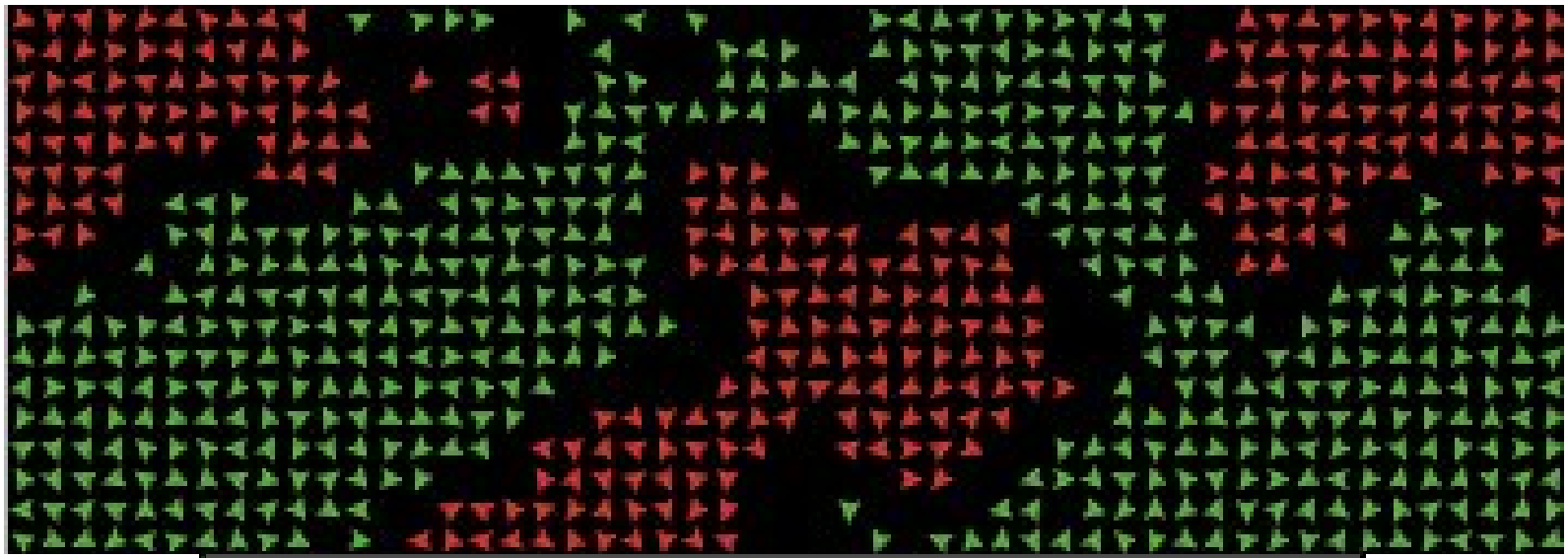
Fourmi de Langton

- Introduit par Christopher Langton en 1986
- Généralités :
  - Une grille unicolore
  - On place une “fourmi” sur une case arbitraire qui peut tourner gauche ou droite et changer les états des cases (les faire passer de blanc à noir et vice-versa)
- Règles :
  - Si sur une case blanche, met la case en noir et va sur la case située à sa droite.
  - Si sur une case noire, met la case en blanc et va sur la case située à sa gauche.



Ségrégation

- Introduit par Sakoda en 1949 et repris par Schelling en 1971
- Généralités :
  - 2 (ou plusieurs) communautés
  - Chaque individu veut vivre à côté d'individu de sa propre communauté
- Règles :
  - Communautés distribuées sur la grille de manière équiprobable sur une grille 2D
  - À chaque membre d'une communauté est donné un même "taux de similarité".
  - À chaque étape de temps, les membres qui ne satisfont pas leur taux de similarité rejoignent une case vide



Feux de forêt

- Généralités :
  - Modéliser la propagation de feux dans les forêts
  - Arbres placés aléatoirement sur une grille
  - Feu lancé sur le côté gauche de la grille
- Règles :
  - Le feu se propage sur les arbres voisins
  - Le voisinage est défini dans le sens de von Neumann
  - Le vent n'est pas pris en compte (s'il n'y a pas d'arbre, le feu s'arrête)