TIPE 2019 : Transport

Numéro candidat : 17535

Automates cellulaires universels

Sommaire

I - Introduction

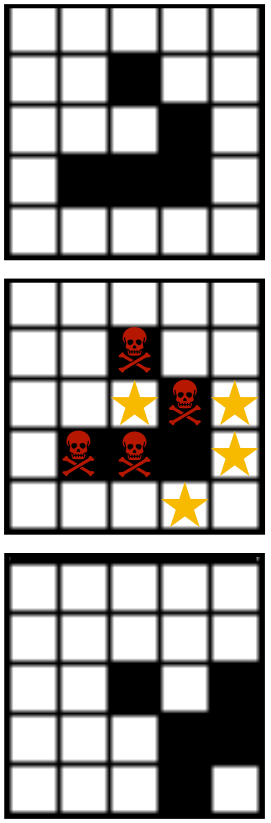
Définitions :

1- Automate cellulaire :

$A = \{\mathbb{Z}^d, Q, V, \delta\}$, avec $\delta : Q^{|V|} \longmapsto Q$

2- Configuration :

Groupe de cellules délimité dans l'espace.
On note C_A l'ensemble des configurations de l'automate A .



Exemple : Jeu de la Vie

d = 2

Q = { □ , ■ } où □ = cellule morte et ■ = cellule vivante

V = $\{v \in \mathbb{Z}^d \mid \| v \|_\infty \leq 1\}$

I - Introduction

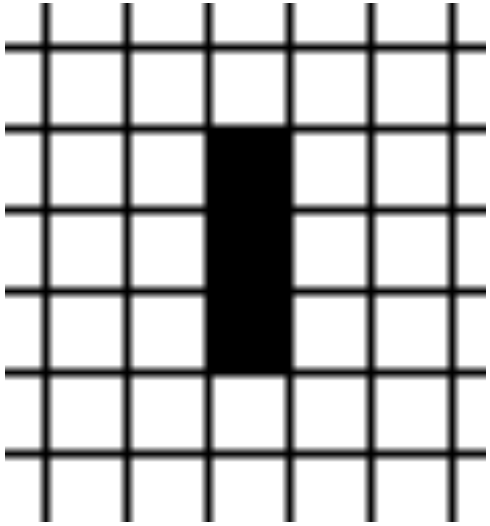
Périodicité :

1- Temporelle

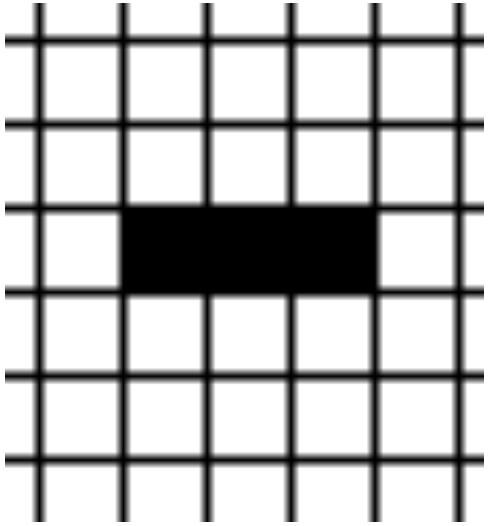
Un automate cellulaire est périodique temporellement si δ est périodique.

Configuration périodique : $C \in C_A$ est périodique temporellement si $\exists t \in \mathbb{N}$ tel que $\delta^t(C) = C$
On note $C_A^{(t)}$ l'ensemble des configurations temporellement périodiques.

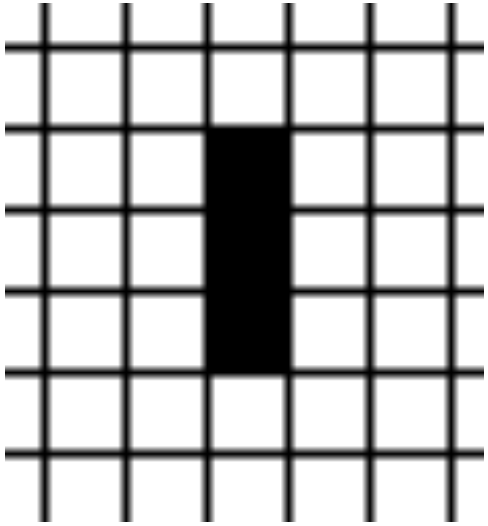
Exemple : Clignotant



Génération 0



Génération 1



Génération 2

I - Introduction

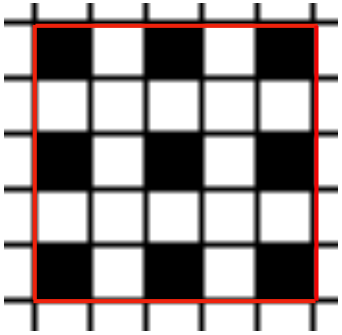
Périodicité :

2- Spatiale :

Une configuration C est spatialement périodique de période π si
$\forall c \in C,\ \forall v \in \{-1,\ 0,\ 1\}^d$ tel que $c + \pi v \in C,\ e_c = e_{c+\pi v}$ où e_c est l'état de la cellule c.
On note $C_A^{(s)}$ l'ensemble des configurations spatialement périodiques.

Un automate cellulaire est spatialement périodique de période π si
$\forall c \in \mathbb{Z}^d,\ \forall v \in \{-1,\ 0,\ 1\}^d,\ e_c = e_{c+\pi v}$

Exemple :



Cette configuration est spatialement périodique de période 2.

I - Introduction

Définition : Un automate cellulaire A est q-nilpotent sur les configurations périodiques si

$$\forall C \in C_A^{(t)} \cup C_A^{(s)},\ \exists n \in \mathbb{N},\ \forall p \geq n,\ \forall e \in \delta_A^p(C),\ e = q$$

II - Universalité

Définitions :

1- Universalité Turing : un automate est dit Turing-universel s'il peut effectuer des calculs algébriques.

2- Universalité intrinsèque : un automate est dit intrinsèque universel s'il peut simuler le comportement de tous les automates cellulaires.

II - Universalité

Simulation du comportement d'un automate cellulaire

Soient deux automates cellulaires A et B de même dimension.

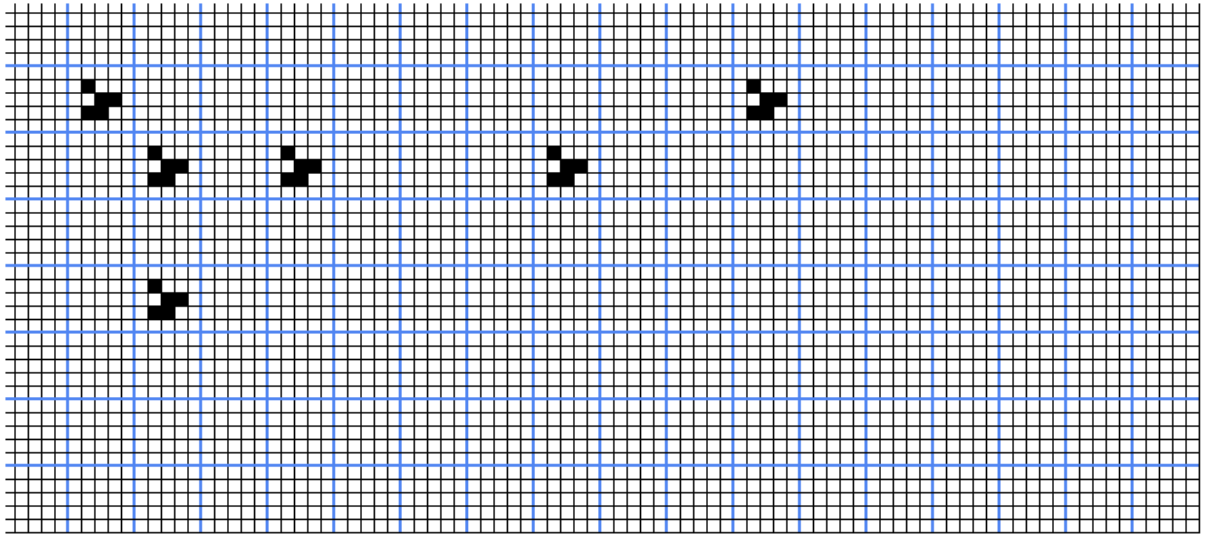
$$A \text{ simule } B \iff \exists m \in \mathbb{N}^* \text{ et } \exists \phi \in Q_B^{Q_A^m},\ \phi \text{ injective, tels que } \exists n \in \mathbb{N} \text{ et} \exists v \in \mathbb{Z}^d \text{ tels que } \overline{\phi} \circ \delta_B = \delta_A^{\langle m,\, n,\, v \rangle} \circ \overline{\phi}$$

Avec $\overline{\phi} : C_A \longmapsto Q_A$

II - Universalité

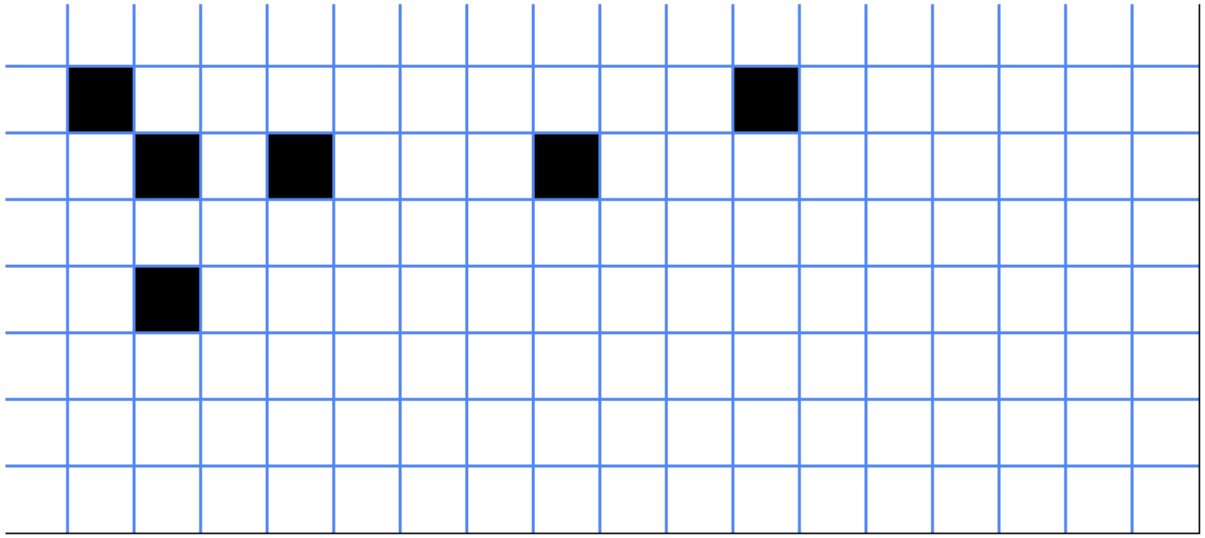
Illustration : Jeu de la Vie simulant l'automate cellulaire « sud-est »

Génération 0



Jeu de la Vie

Génération 0

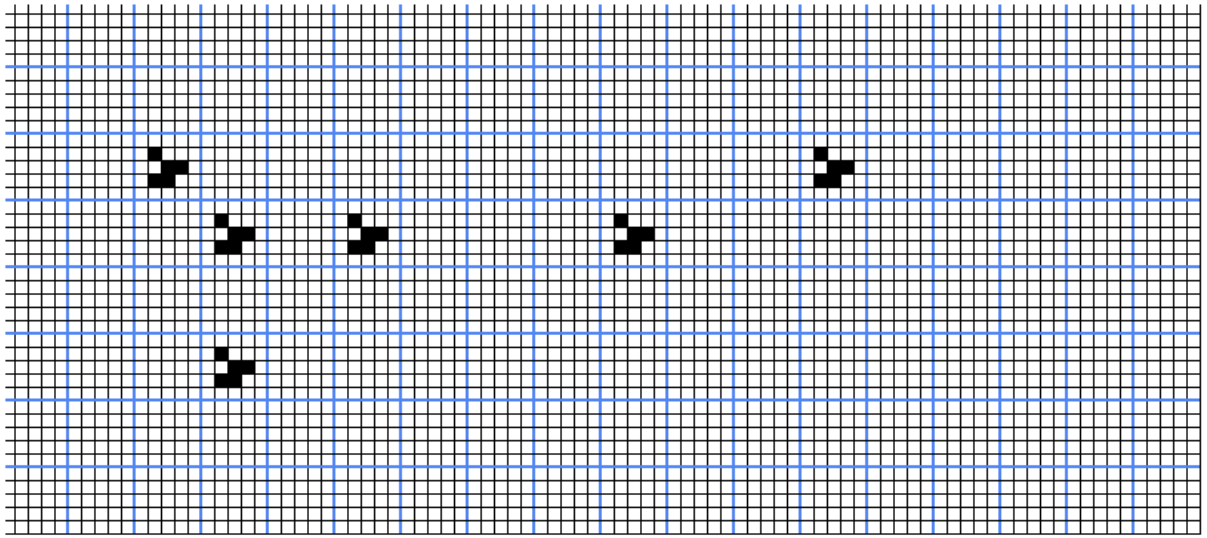


« sud-est »

II - Universalité

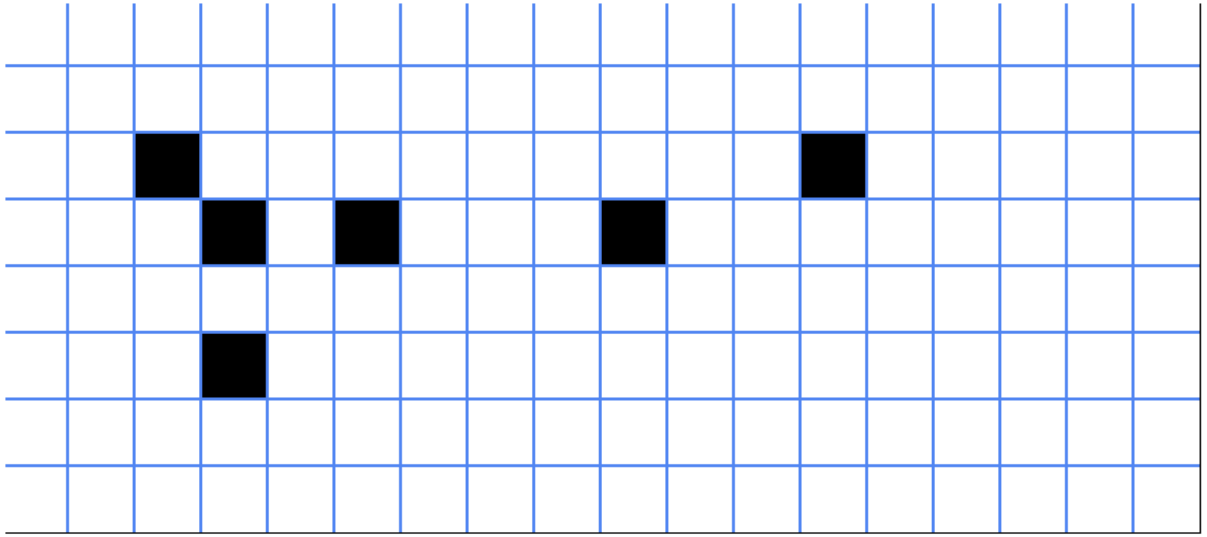
Illustration : Jeu de la Vie simulant l'automate cellulaire « sud-est »

Génération 20



Jeu de la Vie

Génération 1



« sud-est »

III - Preuve de l'indécidabilité de l'universalité

Construction de $\mathbf{U}$

Soit $U = \{\mathbb{Z},\ \{\ \cdot\ \} \cup Q_U,\ V_{vN},\ \delta_U\}$ avec $V_{vN} = \{-1,\ 0\ ,\ 1\}$ le voisinage de Von Neumann.

δ_U est défini sur les configurations localement valides, $\forall (q, q') \in (Q_U)^2,\ \ \| c_q - c_{q'} \|_1 > 1$, et efface les configurations non valides. De plus, δ_U ne crée aucune tête.

Construction de $\mathbf{A \circledast_q U}$

Soit $A \in AC$. Soit $q \in Q_A$.

On définit $A \circledast_q U$ par $A \circledast_q U = \{\mathbb{Z},\ Q_A \times \{\ \cdot\ ,\ \star\ \} \times Q_U,\ V_A \cup V_{UD} \cup V_{vN},\ \delta\}$.

$A \circledast_q U$ est un automate à 3 niveaux dont δ est défini niveau par niveau.

III - Preuve de l'indécidabilité de l'universalité

Niveau inférieur :

Niveau de production d'énergie constitué d'une configuration de A qui évolue selon δ_A et V_A.

Niveau intermédiaire :

Niveau de diffusion d'énergie : ● aucune énergie et ★ une particule énergétique.
Système évoluant comme une translation de vecteur t sauf que si le niveau inférieur ne contient pas q production de ★ dans le niveau intermédiaire.

Niveau supérieur :

Niveau de consommation de l'énergie constitué d'une tête ou d'un blanc de U qui évolue selon δ_U et V_{vN} seulement si le niveau supérieur d'une cellule identifie une tête dans le niveau supérieur d'une de ses voisines et que le niveau intermédiaire possède une particule ★.
Le niveau intermédiaire passe ensuite à l'état sans énergie.

III - Preuve de l'indécidabilité de l'universalité

Lemme : Pour tout automate A , l'automate $A \circledast_q U$ est intrinsèquement universel si et seulement si n'est pas q-nilpotent sur les configurations périodiques.

Démonstration (partielle) :

Supposons A q-nilpotent sur les configurations périodiques.

Soit $B = \{\mathbb{Z}, \{\circ, \bullet\}, V_{UD}, \oplus\}$. Le diagramme espace temps Δ de B représente le triangle de Pascal modulo 2. B est périodique spatialement.

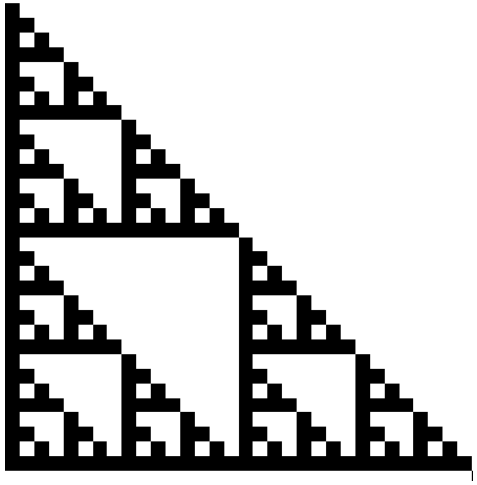


Diagramme élémentaire de B

III - Preuve de l'indécidabilité de l'universalité

Lemme : Pour tout automate A , l'automate $A \circledast_q U$ est intrinsèquement universel si et seulement si n'est pas q-nilpotent sur les configurations périodiques.

Démonstration (partielle) :

Montrons que $\mathbf{A \circledast_q U}$ ne peut pas simuler $\mathbf{B}$.

Supposons $A \circledast_q U$ intrinsèquement universel donc il simule B.

A q-nilpotent sur les configurations périodiques $\Rightarrow$ plus de production d'énergie au bout d'un temps fini.

Seul le niveau intermédiaire agit donc $A \circledast_q U$ est un automate cellulaire de translation de de vecteur t. De plus B est spatialement périodique donc $A \circledast_q U$ se comporte comme un automate cellulaire périodique. Or B ne peut pas être simulé par un tel automate cellulaire.

III - Preuve de l'indécidabilité de l'universalité

Théorème : Le problème de l'universalité intrinsèque pour les automates cellulaires de dimension 1 est indécidable.

Théorème de Jarkko Kari : La q-nilpotence sur les configurations périodiques est indécidable.

IV - Conclusion

V - Annexe

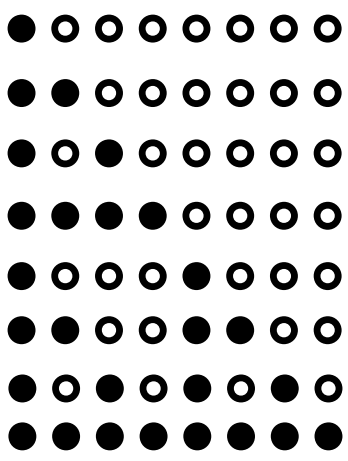
1- Le diagramme élémentaire de $\mathbf{B}$ est le triangle de Pascal modulo 2 :

Soit $(n, p) \in \mathbb{N}^2$, tel que $0 < p < n$, d'après la formule de Pascal,

$$\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$$

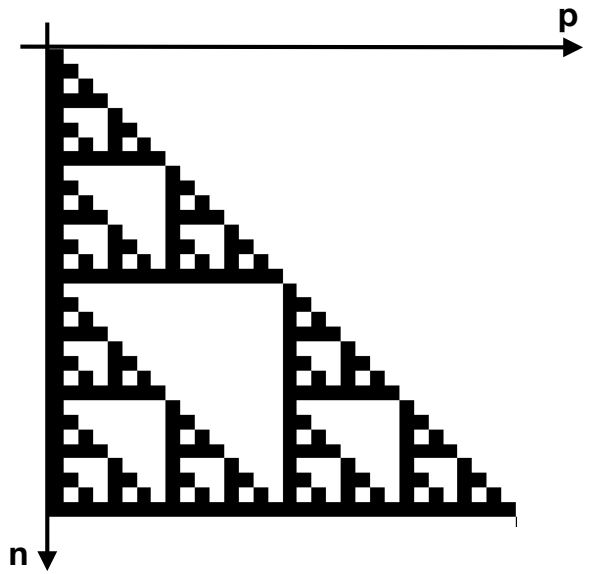
Soit $k \in \mathbb{N}$ si k est pair, on le représente par ∘ ; si k est impair, on le représente par • .

1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	2	1	0	0	0	0	0
1	3	3	1	0	0	0	0
1	4	6	4	1	0	0	0
1	5	10	10	5	1	0	0
1	6	15	20	15	6	1	0
1	7	21	35	35	21	7	1



V - Annexe

2- Preuve de la non simulation de $\mathbf{B}$ par un automate cellulaire périodique :



Considérons $\mathscr{D} = \{(n,p) \in \mathbb{N}^2,\ n \geq p\}$ et $f : (n,p) \in \mathscr{D} \longmapsto \begin{cases} 1, & \binom{n}{p} \equiv 1\ [2] \\ 0, & \binom{n}{p} \equiv 0\ [2] \end{cases}$

Soit $p \in \mathbb{N}$. Posons $g_p : n \in \mathbb{N} \longmapsto f(n,p)$

V - Annexe

2- Preuve de la non simulation de $\mathbf{B}$ par un automate cellulaire périodique :

Montrons que $\forall \mathbf{p} \in \mathbb{N}^*$, $\mathbf{g_p}$ n'est pas p-périodique :

Soit $p \in \mathbb{N}^*$. On a déjà $g_p(p) = 1$. Soit $m \in \mathbb{N}$, $m \geq 2$.

Posons $p = \sum_{i=0}^{m-1} 2^i . p_i$ où $\{p_0, \ldots, p_{m-1}\} \in \{0,1\}^m$. Donc $2.p = \sum_{i=0}^{m-1} 2^{i+1} . p_i$

$$= \sum_{i=1}^{m} 2^i . p_{i-1}$$

D'après le théorème de Lucas, $\binom{2.p}{p} \equiv \prod_{i=1}^{m-1} \binom{p_{i-1}}{p_i} . \binom{0}{p_0} . \binom{p_{m-1}}{0}$ [2]

V - Annexe

2- Preuve de la non simulation de $\mathbf{B}$ par un automate cellulaire périodique :

$$\Longrightarrow \binom{2.p}{p} \equiv \prod_{i=1}^{m-1} \binom{p_{i-1}}{p_i} . \binom{0}{p_0} \ [2]$$

Si p est impair alors $p_0 = 1$ donc $\binom{0}{p_0} = 0$ ainsi $g_p(2.p) = 0 \neq g_p(p)$.

Si p est pair alors $\binom{2.p}{p} \equiv \prod_{i=1}^{m-1} \binom{p_{i-1}}{p_i} \ [2] \equiv \binom{0}{p_1} \dots \binom{p_{m-2}}{p_{m-1}} \ [2]$

Si $p_1 = 1$ alors $\binom{0}{p_1} = 0 \Longrightarrow g_p(2.p) = 0 \neq g_p(p)$

Si $p_1 = 0$ alors $\binom{2.p}{p} \equiv \binom{0}{p_2} \dots \binom{p_{m-2}}{p_{m-1}} \ [2]$

En répétant ce procédé pour tous les p_i, $i \in [\![2, m-2]\!]$, on obtient que g_p est p-périodique si et seulement si $p = 0$ donc g_p n'est pas p-périodique.

V - Annexe

2- Preuve de la non simulation de $\mathbf{B}$ par un automate cellulaire périodique :

Montrons que $\nexists \mathbf{k} \in \mathbb{N}^*$ tel que $\forall (\mathbf{n}, \mathbf{p}) \in \mathscr{D}, \mathbf{f}(\mathbf{n}, \mathbf{p}) = \mathbf{f}(\mathbf{n}+\mathbf{k}, \mathbf{p})$:

Supposons $\exists k \in \mathbb{N}^*$ tel que $\forall (n, p) \in \mathscr{D}, f(n, p) = f(n+k, p)$.

On a alors que pour tout entier naturel p non nul, g_p est k périodique. Or on sait que g_k est k-périodique si et seulement si $k = 0$: contradiction.

Ainsi f n'est pas périodique sur sa première variable donc le triangle de Pascal modulo 2 ne peut être simulé par un automate cellulaire périodique.

V - Annexe

3- Preuve du comportement périodique de $\mathbf{A \circledast_q U}$:

B est spatialement périodique de période k donc $e_c = e_{c+k} \Rightarrow e_c = e_{c+kt}$.

$A \circledast_q U$ se comporte comme un automate cellulaire de translation t donc $e_{c+t} = e_{\delta(c)}$

$$\Rightarrow e_{c+kt} = e_{\delta^k(c)}$$

Ainsi, $e_c = e_{\delta^k(c)}$. Donc $A \circledast_q U$ est k-périodique.

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```python
# -*- coding: iso-8859-1 -*-
from Tkinter import *
import random as rd

def damier1(): #fonction dessinant le tableau
    ligne_vert1()
    ligne_hor1()

def ligne_vert1():
    c_x = 0
    while c_x < width:
        if (c_x//c)%5 == 0:
            can1.create_line(c_x, 0, c_x, height, width =2, fill ='RoyalBlue2')
        else:
            can1.create_line(c_x, 0, c_x, height, width =1, fill ='black')
        c_x+=c

def ligne_hor1():
    c_y = 0
    while c_y < height:
        if (c_y//c)%5 == 0:
            can1.create_line(0, c_y, width, c_y, width =2, fill ='RoyalBlue2')
        else:
            can1.create_line(0, c_y, width, c_y, width =1, fill ='black')
        c_y+=c

def redessiner1(): #fonction redessinant le tableau ?partir de dico_etat
    can1.delete(ALL)
    t=0
    while t!= width//c:
        u=0
        while u!= height//c:
            x=t*c
            y=u*c
            if dico_etat1[x,y]==3:
                dico_case1[x,y]=1
                can1.create_rectangle(x, y, x+c, y+c, fill ='black', width =1)
            elif dico_etat1[x,y]==2:
                if dico_case1[x,y]==1:
                    can1.create_rectangle(x, y, x+c, y+c, fill ='black', width =1)
                else:
                    can1.create_rectangle(x, y, x+c, y+c, fill ='grey98', width =1)
            elif dico_etat1[x,y]<2 or dico_etat1[x,y]>3:
                dico_case1[x,y]=0
                can1.create_rectangle(x, y, x+c, y+c, fill ='grey98', width =1)
            u+=1
        t+=1
    damier1()
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
49
50
51  def damier2(): #fonction dessinant le tableau
52      ligne_vert2()
53      ligne_hor2()
54
55  def ligne_vert2():
56      c_x = 0
57      while c_x < width:
58          can2.create_line(c_x, 0, c_x, height, width =2, fill ='RoyalBlue2')
59          c_x+=5*c
60
61  def ligne_hor2():
62      c_y = 0
63      while c_y < height :
64          can2.create_line(0, c_y, width, c_y, width =2, fill ='RoyalBlue2')
65          c_y+=5*c
66
67  def redessiner2():
68      can2.delete(ALL)
69      t=0
70      while t < width//(5*c):
71          u=0
72          while u < height//(5*c):
73              x=5*t*c
74              y=5*u*c
75              if dico_case2[x,y]==1:
76                  can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black', width =1)
77              else:
78                  can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='grey98', width =1)
79              u+=1
80          t+=1
81      damier2()
82
83
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
84  def detection_planeur(x,y):
85      if dico_case1[x,y+c]==1 and  dico_case1[x+c,y+2*c]==1 and  dico_case1[x+2*c,y]==1 and  dico_case1[x+2*c,y+1*c]==1 and  dico_case1[x+2*c,y+2*c]==1
86      or dico_case1[x,y+2*c]==1 and dico_case1[x+c,y+3*c]==1 and  dico_case1[x+2*c,y+c]==1 and  dico_case1[x+2*c,y+2*c]==1 and  dico_case1[x+2*c,y+3*c]==1
87      or dico_case1[x,y+3*c]==1 and  dico_case1[x+c,y+4*c]==1 and dico_case1[x+2*c,y+2*c]==1 and  dico_case1[x+2*c,y+3*c]==1 and  dico_case1[x+2*c,y+4*c]==1
88      or dico_case1[x+c,y+c]==1 and  dico_case1[x+2*c,y+2*c]==1 and  dico_case1[x+3*c,y]==1 and dico_case1[x+3*c,y+1*c]==1 and  dico_case1[x+3*c,y+2*c]==1
89      or dico_case1[x+c,y+2*c]==1 and  dico_case1[x+2*c,y+3*c]==1 and  dico_case1[x+3*c,y+c]==1 and  dico_case1[x+3*c,y+2*c]==1 and  dico_case1[x+3*c,y+3*c]==1
90      or dico_case1[x+c,y+3*c]==1 and  dico_case1[x+2*c,y+4*c]==1 and  dico_case1[x+3*c,y+2*c]==1 and  dico_case1[x+3*c,y+3*c]==1 and  dico_case1[x+3*c,y+4*c]==1
91      or dico_case1[x+2*c,y+c]==1 and  dico_case1[x+3*c,y+2*c]==1 and  dico_case1[x+4*c,y]==1 and  dico_case1[x+4*c,y+1*c]==1 and  dico_case1[x+4*c,y+2*c]==1
92      or dico_case1[x+2*c,y+2*c]==1 and  dico_case1[x+3*c,y+3*c]==1 and  dico_case1[x+4*c,y+c]==1 and  dico_case1[x+4*c,y+2*c]==1 and  dico_case1[x+4*c,y+3*c]==1
93      or dico_case1[x+2*c,y+3*c]==1 and  dico_case1[x+3*c,y+4*c]==1 and  dico_case1[x+4*c,y+2*c]==1 and  dico_case1[x+4*c,y+3*c]==1 and  dico_case1[x+4*c,y+4*c]==1:
94          dico_case2[x,y]=1
95          return True
96      else :
97          dico_case2[x,y]=0
98          return False
99
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```python
def planeur():
    x,y=5*c,5*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')
    dico_case1[x,y+2*c]=1 ; can1.create_rectangle(x, y+2*c, x+c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+2*c]=1 ; can1.create_rectangle(x+c, y+2*c, x+2*c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')
    x,y=10*c,10*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')
    dico_case1[x,y+2*c]=1 ; can1.create_rectangle(x, y+2*c, x+c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+2*c]=1 ; can1.create_rectangle(x+c, y+2*c, x+2*c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')
    x,y=20*c,10*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')@
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')
    x,y=40*c,10*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')
    dico_case1[x,y+2*c]=1 ; can1.create_rectangle(x, y+2*c, x+c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+2*c]=1 ; can1.create_rectangle(x+c, y+2*c, x+2*c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')
    x,y=55*c,5*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')
    dico_case1[x,y+2*c]=1 ; can1.create_rectangle(x, y+2*c, x+c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+2*c]=1 ; can1.create_rectangle(x+c, y+2*c, x+2*c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')
    x,y=10*c,20*c
    dico_case1[x,y]=1 ; can1.create_rectangle(x, y, x+c, y+c, fill ='black')
    dico_case1[x,y+2*c]=1 ; can1.create_rectangle(x, y+2*c, x+c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+2*c]=1 ; can1.create_rectangle(x+c, y+2*c, x+2*c, y+3*c, fill ='black')
    dico_case1[x+1*c,y+1*c]=1 ; can1.create_rectangle(x+c, y+c, x+2*c, y+2*c, fill ='black')
    dico_case1[x+2*c,y+c]=1 ; can1.create_rectangle(x+2*c, y+c, x+3*c, y+2*c, fill ='black')
    can2.create_rectangle(x, y, x+5*c, y+5*c, fill ='black')

```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
142 def effacer():
143     stop()
144     i=0
145     while i!= width//c:
146         j=0
147         while j!= height//c:
148             x=i*c
149             y=j*c
150             dico_case1[x,y]=0
151             if j%5 == 0 and i%5 == 0:
152                 dico_case2[x,y]=0
153             j+=1
154         i+=1
155     can1.delete(ALL)
156     damier1()
157     can2.delete(ALL)
158     damier2()
159 
160 def click_gauche(event):
161 #fonction rendant vivante la cellule cliquée donc met la valeur 1 pour cette cellule au dico_case1
162     x = event.x -(event.x%c)
163     y = event.y -(event.y%c)
164     can1.create_rectangle(x, y, x+c, y+c, fill ='black')
165     dico_case1[x,y]=1
166 
167 def click_droit(event):
168 #fonction tuant la cellule cliquée donc met la valeur 0 pour la cellule cliquée au dico_case1
169     x = event.x -(event.x%c)
170     y = event.y -(event.y%c)
171     can1.create_rectangle(x, y, x+c, y+c, fill ='white')
172     dico_case1[x,y]=0
173 
174 def change_vit(event): #fonction pour changer la vitesse (l'attente entre chaque étape)
175     global vitesse
176     vitesse = int(eval(entree.get()))
177     print(vitesse)
178 
179 def go():
180     "demarrage de l'animation"
181     global flag
182     if flag ==0:
183         flag =1
184         play()
185 
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
186 def stop():
187     "arret de l'animation"
188     global flag
189     flag =0
190 
191 
192 def play(): #fonction comptant le nombre de cellules vivantes autour de chaque cellule
193     global flag, vitesse
194     v=0
195     while v!= width//c:
196         w=0
197         while w!= height//c:
198             x=v*c
199             y=w*c
200 
201             #cas spéciaux:
202             #les coins
203             if x==0 and y==0: #coin en haut à gauche
204                 compt_viv=0
205                 if dico_case1[x, y+c]==1:
206                     compt_viv+=1
207                 if dico_case1[x+c, y]==1:
208                     compt_viv+=1
209                 if dico_case1[x+c, y+c]==1:
210                     compt_viv+=1
211                 dico_etat1[x, y]=compt_viv
212             elif x==0 and y==int(height-c): #coin en bas à gauche
213                 compt_viv=0
214                 if dico_case1[x, y-c]==1:
215                     compt_viv+=1
216                 if dico_case1[x+c, y-c]==1:
217                     compt_viv+=1
218                 if dico_case1[x+c, y]==1:
219                     compt_viv+=1
220                 dico_etat1[x, y]=compt_viv
221             elif x==int(width-c) and y==0: #coin en haut à droite
222                 compt_viv=0
223                 if dico_case1[x-c, y]==1:
224                     compt_viv+=1
225                 if dico_case1[x-c, y+c]==1:
226                     compt_viv+=1
227                 if dico_case1[x, y+c]==1:
228                     compt_viv+=1
229                 dico_etat1[x, y]=compt_viv
230             elif x==int(width-c) and y==int(height-c): #coin en bas à droite
231                 compt_viv=0
232                 if dico_case1[x-c, y-c]==1:
233                     compt_viv+=1
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
                if dico_case1[x-c, y]==1:
                    compt_viv+=1
                if dico_case1[x, y-c]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv

            #cas spéciaux:
            #les bords du tableau (sans les coins)
            elif x==0 and 0<y<int(height-c): #bord de gauche
                compt_viv=0
                if dico_case1[x, y-c]==1:
                    compt_viv+=1
                if dico_case1[x, y+c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y]==1:
                    compt_viv+=1
                if dico_case1[x+c, y+c]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv
            elif x==int(width-c) and 0<y<int(height-c): #bord de droite
                compt_viv=0
                if dico_case1[x-c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x-c, y]==1:
                    compt_viv+=1
                if dico_case1[x-c, y+c]==1:
                    compt_viv+=1
                if dico_case1[x, y-c]==1:
                    compt_viv+=1
                if dico_case1[x, y+c]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv
            elif 0<x<int(width-c) and y==0: #bord du haut
                compt_viv=0
                if dico_case1[x-c, y]==1:
                    compt_viv+=1
                if dico_case1[x-c, y+c]==1:
                    compt_viv+=1
                if dico_case1[x, y+c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y]==1:
                    compt_viv+=1
                if dico_case1[x+c, y+c]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv
            elif 0<x<int(width-c) and y==int(height-c): #bord du bas
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
                compt_viv=0
                if dico_case1[x-c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x-c, y]==1:
                    compt_viv+=1
                if dico_case1[x, y-c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv

            #cas généraux
            #les cellules qui ne sont pas dans les bords du tableau
            else:
                compt_viv=0
                if dico_case1[x-c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x-c, y]==1:
                    compt_viv+=1
                if dico_case1[x-c, y+c]==1:
                    compt_viv+=1
                if dico_case1[x, y-c]==1:
                    compt_viv+=1
                if dico_case1[x, y+c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y-c]==1:
                    compt_viv+=1
                if dico_case1[x+c, y]==1:
                    compt_viv+=1
                if dico_case1[x+c, y+c]==1:
                    compt_viv+=1
                dico_etat1[x, y]=compt_viv

            if x%(5*c)==0 and y%(5*c)==0:
                if detection_planeur(x,y):
                    redessiner2()
            w+=1
        v+=1
    redessiner1()
    if flag >0:
        fen1.after(vitesse,play)
```

V - Annexe

Illustration_simulation_GOL_SE_tkinter.py

```
#les différentes variables:

#taille des cellules
c = 9

#taille de la grille
height = 40*c
width = 90*c

#vitesse de l'animation
vitesse=1000

flag=0
dico_etat1 = {} #dictionnaire contenant le nombre de cellules vivantes autour de chaque cellule
dico_case1, dico_case2 = {}, {} #dictionnaire contenant les coordonnées de chaques cellules et
                                #une valeur 0 ou 1 si elles sont respectivement mortes ou vivantes
i=0
while i!= width//c:
    j=0
    while j!= height//c:
        x=i*c
        y=j*c
        dico_case1[x,y]=0
        if j%5 == 0 and i%5 == 0:
            dico_case2[x,y]=0
        j+=1
    i+=1


#programme "principal"
fen1 = Tk()

can1 = Canvas(fen1, width =width, height =height, bg ='white')
can1.bind("<Button-1>", click_gauche)
can1.bind("<Button-2>", click_droit)
can1.pack(side =TOP, padx =5, pady =5)

damier1()

can2 = Canvas(fen1, width =width, height =height, bg ='white')
can2.pack(side =TOP, padx =5, pady =5)

damier2()

```

V - Annexe

```
372
373   b1 = Button(fen1, text ='Play', command =go)
374   b2 = Button(fen1, text ='Stop', command =stop)
375   b3 = Button(fen1, text ='Planeur', command =planeur)
376   b4 = Button(fen1, text ='Effacer', command =effacer)
377   b1.pack(side =LEFT, padx =3, pady =3)
378   b2.pack(side =LEFT, padx =3, pady =3)
379   b3.pack(side =LEFT, padx =3, pady =3)
380   b4.pack(side =LEFT, padx =3, pady =3)
381
382   entree = Entry(fen1)
383   entree.bind("<Return>", change_vit)
384   entree.pack(side =RIGHT)
385   chaine = Label(fen1)
386   chaine.configure(text ="Attente entre chaque ?ape (ms) :")
387   chaine.pack(side =RIGHT)
388
389
390   fen1.mainloop()
391
```