NaN – Défi n°5 – Automates cellulaires

Attention : je ne suis pas enseignant et je n'ai pas suivi une formation académique. Ce document ne doit en aucun cas être considéré comme un cours ! 😉

Introduction

On imagine un « monde » constitué d'une **grille peuplée de cellules**. L'état de chaque cellule à l'instant $t+1$ est donné par l'évaluation d'une fonction **sur les cellules voisines** à l'instant t. Cette fonction est nommée « fonction de transition ».

On appelle « automate cellulaire » un programme capable de calculer les états successifs de la grille-monde.

En faisant varier la dimension, la taille ou le degré de connexité du monde (c'est-à-dire la forme des cellules), ainsi que le voisinage ou la fonction de transition, les automates cellulaires donnent naissance à des objets **très riches** et **très différents**.

Une formulation équivalente (si ça vous effraie, sautez !)

On nomme $E = \{e_0, e_1, \dots, e_{n-1}\}$ l'ensemble des n états possibles pour chaque cellule du monde, et $W = \{w_i | i \in \mathbb{N}^m, w_i \in E\}$ le monde de dimension m (c'est-à-dire l'ensemble des cellules w).

Soit $v: W \to E^k$ la fonction *voisinage*, qui permet d'obtenir l'état des k cellules voisines d'une certaine cellule du monde, et $f: (E, E^k) \to E$ la fonction de transition.

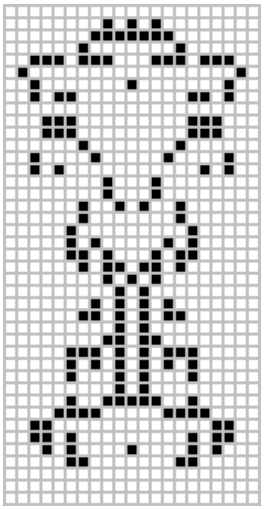
L'état du monde W à l'instant $t+1$ est alors :

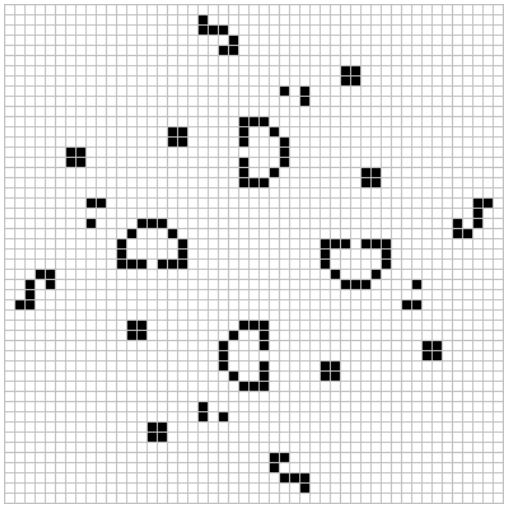
$$W_{t+1} = \{f(w_i, v(w_i)) | w_i \in W_t\}$$

On remarque que f et v sont homogènes sur le monde, et que l'évolution du monde dépend du choix de la fonction de voisinage.

L'exemple canonique : le jeu de la vie de John Conway

Le plus célèbre des automates cellulaires est probablement le « jeu de la vie », formulé par le mathématicien John Conway en 1970.





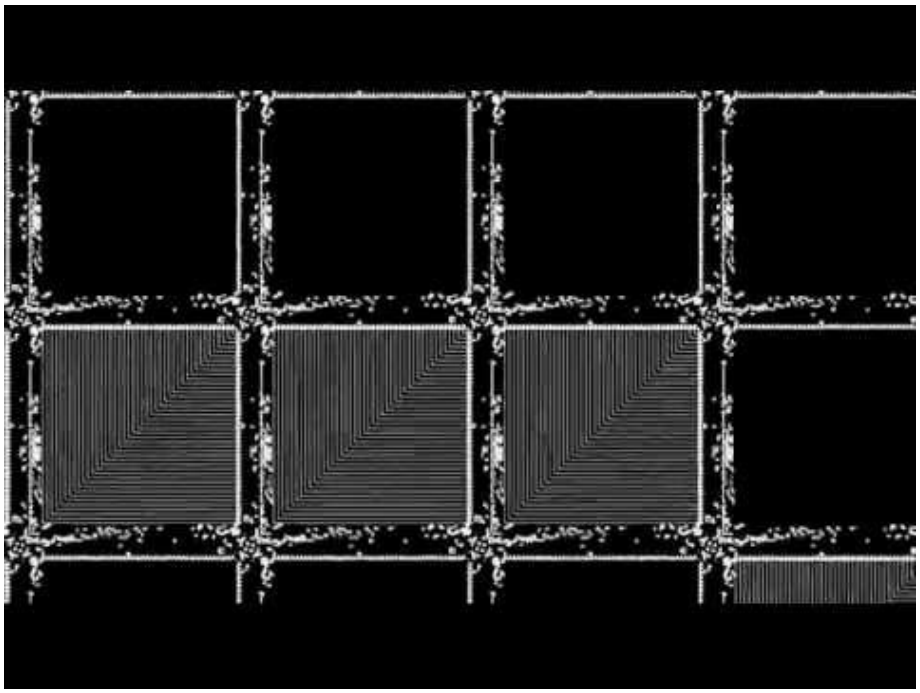
Deux structures dans Life*, l'une mobile, l'autre statique. Cliquez sur les images pour les voir sous forme animée !*

Les « règles » de cet automate sont les suivantes :

- Le monde est une **grille de dimension 2** et de taille arbitraire, où chaque cellule est carrée ;
- Chaque cellule peut avoir **deux états**, vivante ou morte ;
- Le voisinage d'une cellule est composé des **8 cellules adjacentes** ;
- Enfin, la fonction de transition est la suivante :
    - Une cellule **morte** possédant **exactement trois** voisines vivantes devient **vivante** (les cellules se « reproduisent ») ;
    - Une cellule **vivante** ne possédant **pas exactement deux ou trois** voisines vivantes devient **morte** (la cellule meurt de « solitude » ou de « surpopulation »).

On représente classiquement les cellules vivantes comme des carrés noir, les cellules mortes comme des carrés blancs. En appliquant successivement la fonction de transition à l'ensemble des cellules du monde, des motifs très riches et des comportements fascinants émergent de ces règles, pourtant très simples.

Life possède une assez grande communauté de « joueurs », qui construisent des structures parfois gigantesques, capables de se déplacer, de se cloner, de transmettre des données…



Life in Life*, un automate cellulaire construit dans* Life *lui-même, exprime la Turing-complétude de* Life*.*

Un extra : formulation mathématique de la fonction de transition…

On définit les cellules vivantes comme à l'état logique *vrai* (V) et les cellules mortes comme à l'état logique *faux* ($\bar{V}$). En appelant E l'état de la cellule actuelle et x le nombre de cellules voisines vivantes, on peut définir la fonction de transition comme suit :

$$f \coloneqq \begin{cases} (E, x) \to (x = 3) \vee \big(E \wedge (x = 2)\big) \\ (\{V, \bar{V}\}, [\![0, 8]\!]) \to \{V, \bar{V}\} \end{cases}$$

Le défi !

Vous êtes fin prêt à réaliser votre propre automate cellulaire !

En fonction de votre niveau, des capacités des outils que vous désirez utiliser, et du temps que vous souhaitez y consacrer, **trois niveaux** de complexité sont proposés. Le troisième niveau, plus libre, sera sujet à un **vote de la communauté** qui élira la proposition la plus intéressante ! ✨

Vous êtes évidemment libre de réaliser les niveaux qu'il vous plaira, merci toutefois de ne soumettre qu'**une seule proposition**.

Niveau 1 : Un automate monodimensionnel totalistique

Ne vous laissez pas impressionner par le nom, c'est bien moins compliqué que ça en a l'air ! 😛

Contrairement au jeu de la vie qui se déroule dans un monde en deux dimensions, le monde de l'automate que vous devez réaliser ne comporte **qu'une seule ligne** de cellules, il ne se « joue » que sur une seule dimension.

Dans ce monde, les cellules peuvent prendre **quatre états**, correspondant aux nombres $\{0,1,2,3\}$.

0	2	0	1	2	3	3	0	1	1

Une vue d'un monde correspondant à ces règles, à un état quelconque.

Et la fonction de transition ? Elle est un peu différente de celle de *Life*, mais elle est en fait plus simple. En effet, **le nouvel état d'une cellule dépendra de la somme de la cellule et de ses deux voisines**, selon le tableau suivant :

Valeur de la somme	0	1	2	3	4	5	6	7	8	9
Nouvel état	0	3	2	0	0	1	3	2	3	1

Pas clair ? 😅 Bon, un exemple. Reprenons le monde du paragraphe précédent, et supposons que je veuille obtenir le nouvel état de la cellule colorée en bleu :

0	2	0	1	2	3	3	0	1	1

Pour cela, je fais la somme entre la cellule et son voisinage, coloré en gris : $0 + 1 + 2 = 3$. D'après le tableau de la fonction de transition, si la somme vaut 3, le nouvel état de la cellule sera 0.

Votre mission est donc de faire un programme qui **calcule l'évolution de ce monde**, et qui affiche ses états successifs. Vous pouvez par exemple afficher ces états ligne par ligne dans un terminal, où chaque valeur de cellule est représentée par un caractère ; ou bien générer une image dont chaque ligne de pixels correspond à un état du monde.

Quelques conseils :

- Un très bon état initial pour votre monde est une ligne remplie de zéros, avec **une unique cellule à l'état 1 au centre**. Son évolution pourrait vous surprendre…

…	0	0	0	1	0	0	0	…

Un état initial vivement recommandé !

- Si c'est compliqué, ne perdez pas trop de temps avec un affichage graphique ! Vous pouvez obtenir de très bons résultats dans le terminal, essayez par exemple de représenter les nombres $\{0,1,2,3\}$ par les caractères « », « . », « | » et « X ».

Niveau 2 : Un simulateur de feu de forêt

Pour ce simulateur, le monde est une **grille bidimensionnelle** de taille arbitraire (identique au jeu de la vie).

Les cellules peuvent être dans les états suivant. Selon la manière dont vous souhaitez afficher votre résultat (image, terminal…), un code couleur et des emojis sont également suggérés :

- Cendres 💀 (noir)
- Forêt jeune 🌱 (vert clair)
- Forêt ancienne 🌳 (vert foncé)
- Début de combustion ✨ (jaune)
- En combustion 🔥 (rouge)
- Fin de combustion 🧯 (orange)

On considère le voisinage d'une cellule comme étant **ses 8 cellules adjacentes**. Les règles de transition sont les suivantes (les règles les plus hautes dans le tableau sont prioritaires) :

Ancien état	**Nouvel état**	**Condition sur le voisinage**	**Proba.**
Forêt jeune	Début de comb.	Une voisine au moins en début de combustion	1%
		Une voisine au moins en combustion	2%
		Une voisine au moins en fin de combustion	1%
Forêt ancienne	Début de comb.	Une voisine au moins en début de combustion	10%
		Une voisine au moins en combustion	20%
		Une voisine au moins en fin de combustion	10%
Début de comb.	Combustion	*Aucune*	10%
Combustion	Fin de comb.		
Fin de comb.	Cendres		
Cendres	Forêt jeune	*Aucune*	0,1%
Forêt jeune	Forêt ancienne		0,5%
Forêt ancienne	Début de comb.	Cinq voisines au moins en forêt ancienne	0,005%

Contrairement aux automates proposés jusqu'ici, celui-ci introduit la notion de hasard. Par exemple, si une cellule contient de la forêt ancienne **et** qu'une de ses voisines au moins est en combustion, la cellule à 20% de chances de s'embraser à son tour.

Vous pouvez initialiser le monde comme vous le souhaiter, par exemple avec toutes les cellules à l'état « cendres » si vous voulez voir votre forêt pousser, ou avec toutes les cellules à l'état « forêt jeune » si vous êtes pressés de voir vos premiers incendies. 😊

Niveau 3 : Mini-concours, définissez les règles du jeu !

En vous inspirant des règles présentées ici ou en créant de nouvelles de toutes pièces, **fabriquez un automate cellulaire** original.

Laissez libre cours à votre imagination et **expérimentez** : voisinages asymétriques, 3D, grilles triangles ou hexagonales, règles de transition étranges… Tout est possible ! Notez également que rien n'oblige les cellules à se retreindre à un nombre fini d'états, ou à n'être que dans un seul état à la fois. 😄

Peut-être est-ce purement abstrait, ou peut-être allez-vous partir d'un phénomène réel ? Peut-être est-ce interactif, avec une interface graphique ? Ou bien uniquement affiché dans un terminal ? 😊

Une fois satisfait de votre automate, réalisez-en une capture (image fixe, GIF, vidéo…). Les membres de NaN seront invités à **voter pour celui qu'ils préfèrent** !

Astuce : comme l'illustre le jeu de la vie, ***il n'y a pas besoin de partir dans des choses compliquées pour obtenir des résultats très intéressants****. Commencez par fixer des paramètres et des règles simples, et complexifiez-les au fur et à mesure, plutôt que de créer d'une traite un jeu de règles compliqué que vous aurez du mal à faire évoluer.*