Automates cellulaires

Le projet est à réaliser en **binôme**. Le rendu devra se faire avant le **Mardi 01 Octobre 2019 — 23h59** sur CELENE exclusivement, et devra comprendre **exactement** deux fichiers :

- le code source (et tous les fichiers nécessaires à l'éxécution) **commenté** et correctement archivé au format `tar.gz`.
- un rapport (trois pages maximum) au format PDF, expliquant la structure de votre code **et** son utilisation.

Les fichiers à rendre

L'archive devra contenir **uniquement** les fichiers source, ainsi que tous les éléments nécessaires à la compilation (*e.g.* `Makefile`) **et** à l'exécution.

Introduction

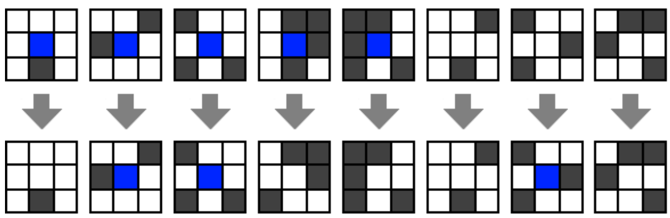
Le contexte.

Un automate cellulaire est un ensemble de **cellules** contenant chacune un **état**, choisi parmi un ensemble fini et pouvant évoluer au cours du temps. L'état d'une cellule au temps $t + 1$ est décidé en fonction de l'état de son voisinage (ensemble fini de cellules) au temps t. Ces règles sont appliquées **simultanément** sur l'ensemble des cellules, produisant une génération de cellules dépendant entièrement de la précédente.

(a) Voisinage de MOORE.

(b) Voisinage de VON NEUMANN.

FIGURE 1 – Voisinages standards pour des automates à 2 dimensions, contenant respectivement 8 et 4 cellules.



(a) **Jeu de la vie** et voisinage de MOORE.

Une dimension, deux états

That's one state one state two state.

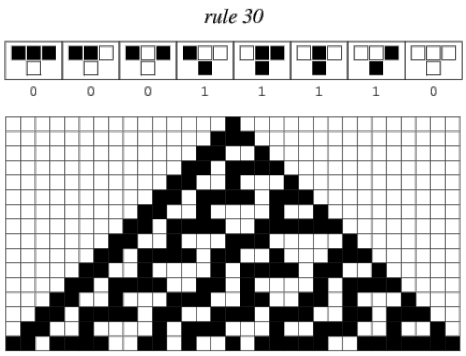
Un automate cellulaire à une dimension est représenté par une grille **unidimensionnelle**, **affichant** l'état de toutes les cellules à l'instant t. Une représentation possible de l'évolution d'un automate cellulaire à une dimension est la suivante :

— situer l'instant 0 (appelé **configuration initiale**) en haut de la représentation
— étant donnée une configuration C_t obtenue au temps t, la ligne suivante contient la configuration C_{t+1}.

Le **voisinage** d'une cellule contient la cellule elle-même, ainsi que les cellules se trouvant à ses côtés (gauche et droite).

Automates cellulaires élémentaires (à deux états)

Le nombre de configurations possibles pour un voisinage donné est de 2^3, et il est donc possible d'établir $2^{2^3} = 256$ automates cellulaires, dits **élémentaires**. Le **code de Wolfram** propose d'attribuer à chacun de ces ensembles de règles un **nombre** compris entre 0 et 255. Plus précisément, les voisinages possibles sont considérés dans l'ordre $111, 110, \ldots, 001, 000$ et à chaque triplet correspond un état (0 ou 1). Une **règle** peut donc être vue comme la valeur binaire représentée par les états successifs.



La figure ci-contre représente ainsi la règle :

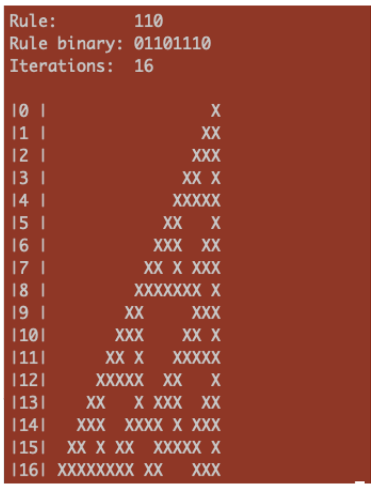
$$(00011110)_2 = 30$$

Représentation versus Implémentation. La configuration C_t d'un automate cellulaire est suffisante pour obtenir la configuration suivante C_{t+1}. Un automate sera donc **affiché** sous forme de grille, mais pourra être **implémenté** différemment.

$\mathcal{Q}_1$. Proposer une **structure** et un code permettant de **simuler** un automate cellulaire élémentaire. Le code devra permettre le calcul des configurations successives d'un automate cellulaire à une dimension, ainsi que son affichage en console.

Structure et règles binaires

Instructions. Les états seront représentés par des `char`. Afin d'obtenir un affichage cohérent, le nombre d'itérations sera borné (par ex. 16) et la dimension de l'automate sera également limitée (par ex. 32). La configuration initiale et la règle de transition pourront être saisies en dur dans votre fonction principale, mais une gestion par argument sera grandement appréciée. La sortie produite par votre code pourra ressembler à l'image de droite.



$\mathcal{Q}_1$. Pour permettre une meilleure modularité du code, l'affichage devrait être **générique**. Modifier le code précédent pour que l'affichage soit géré par une **fonction** qui pourra être intégrée à la structure de votre code.

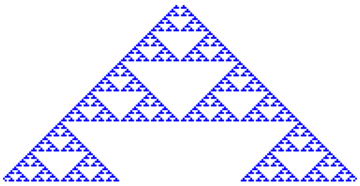
Généricité

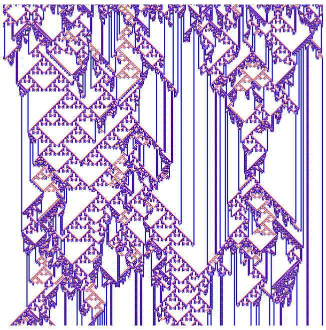
États supplémentaires *That's two states two states speed bikes.*

Nous nous intéressons maintenant à des automates cellulaires à une dimension contenant **quatre** états, représentés par les entiers $\{0, 1, 2, 3\}$. Les règles de transition sont définies en fonction de la **somme** du voisinage de la cellule courante (valeur comprise entre 0 et 9). Une manière de représenter ces règles consiste à fournir un **tableau de 10 éléments**, ayant des valeurs comprises entre 0 et 3. La somme du voisinage d'une cellule donnera ainsi un **indice** du tableau, qui contiendra le nouvel état de la cellule.

The big Sierpinski

Considérons la règle 0100000000, qui signifie qu'une cellule passe dans l'état 1 lorsque la somme de son voisinage vaut 1, et 0 sinon. On obtient alors le triangle de Sierpinski (voir au milieu). L'image de droite représente la règle 0013100132 (les états étant colorés). Les images sont extraites de ce site.





Q_1. Adapter le code précédent pour permettre la gestion d'un nombre d'états arbitraire et la règle de transition **somme**.

Règles supplémentaires

Remarque. A nouveau, afin d'avoir un code modulable, il serait bon de pouvoir appliquer **n'importe quelle** règle de transition sur l'automate cellulaire.

Extensions *Sunset disco this if for you.*

Q_1. Adapter le code pour permettre la gestion de la configuration initiale et de la règle de transition depuis un fichier. Le format de ce dernier est laissé libre mais **doit** être justifié dans le rapport.

Gestion de fichiers

Q_1. En utilisant la librairie `pgm_img` du TP5[a], créer une fonction permettant d'enregistrer l'automate cellulaire dans une image `PGM`. Cette fonction pourra être passée en paramètre de l'affichage, pour obtenir une image à la place d'un affichage console.

Généricité de la vue

a. Une version codée lors de ce TP est préférable, mais une version sera proposée sur CELENE environ à mi-parcours.