Informatique (INFO-F206) – Projet – Automates Cellulaires

Jean Cardinal

8 novembre 2011

1 Introduction

Un automate cellulaire est un tableau de *cellules* qui évolue avec le temps selon une règle *locale* et *déterministe*. Une cellule peut être dans un certain nombre d'états, typiquement *morte* ou *vivante*, respectivement représentés par les nombres 0 et 1. À chaque étape d'évolution, une cellule peut changer d'état, suivant l'état des cellules voisines dans le tableau. L'état de la cellule à l'étape suivante est déterminé par une règle simple. Dans le cas que nous traiterons, des automates cellulaires *unidimensionnels*, le tableau est un tableau à une dimension, et chaque indice dans ce tableau correspond à une cellule.

Règle de transition. Un exemple de règle de transition est le suivant :

État précédent			Nouvel état
Indice $i-1$	Indice i	Indice $i+1$	Indice i
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

Cette règle est appelée la **règle 30**, car les chiffres de la représentation binaire du nombre 30 correspondent aux résultats pour chaque voisinage dans cet ordre : $30 = 00011110_2$. La règle est appliquée simultanément à toutes les cellules à une étape donnée et donne les états des cellules à l'étape suivante.

La **règle 60** peut être décrite comme suit :

État précédent			Nouvel état
Indice $i-1$	Indice i	Indice $i+1$	Indice i
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

On vérifie que le nombre binaire 00111100_2 s'écrit bien 60 en base 10.

Conditions aux bords. L'état de la cellule i à l'étape suivante dépend de son état, ainsi que de l'état de ses deux voisines, d'indices $i-1$ et $i+1$. Cependant, la première et la dernière cellule, d'indices respectifs 0 et $n-1$, où n est la taille du tableau, ne possèdent qu'une seule voisine. Deux solutions sont

envisageables : on peut considérer que les voisines inexistantes sont toujours *mortes* (dans l'état 0), ou considérer le tableau de manière circulaire, de façon que la voisine gauche de la cellule d'indice 0 est la cellule d'indice $n-1$, et la voisine droite de la cellule $n-1$ est la cellule d'indice 0. C'est cette dernière solution que nous retiendrons.

Exemple. Voici le résultat obtenu pour 5 applications successives de la règle 30, en partant du tableau `0000010000000000` :

```
0000010000000000
0000111100000000
0001100100000000
0011011110000000
0110010001000000
1101111011100000
```

Et voici le résultat pour 19 applications de la règle 60, en partant du tableau `00000010000000000000` :

```
00000010000000000000
00000011000000000000
00000010100000000000
00000011110000000000
00000010001000000000
00000011001100000000
00000010101010000000
00000011111111000000
00000010000000100000
00000011000000110000
00000010100000101000
00000011110000111100
00000010001000100010
00000011001100110011
10000010101010101010
11000011111111111111
00100010000000000000
00110011000000000000
00101010100000000000
00111111110000000000
```

2 Énoncé du travail

On vous demande d'écrire un petit programme permettant de visualiser l'évolution d'un automate cellulaire unidimensionnel de taille quelconque, en utilisant une règle de transition de manière circulaire. Pour cela, vous devrez utiliser les structures du langage Python vues en cours. En particulier, on vous demande :

- de diviser le programme en fonctions élémentaires, traitant respectivement de :
  - l'application d'une règle à une cellule
  - l'application d'une règle à toutes les cellules d'un tableau
  - l'application itérative d'une règle à un tableau de départ, avec sauvegarde et/ou affichage des états successifs,
- de commenter de manière pertinente les différentes parties du programme (notamment en utilisant les « `docstrings` », voir syllabus, Chapitre 3, « Documenter ses fonctions »),
- d'utiliser les paramètres en ligne de commande, comme décrit ci-dessous.

Paramètres en ligne de commande. L'interpréteur Python peut être utilisé de différentes manières. La manière la plus simple est d'utiliser l'outil IDLE (cf. cours). Il est cependant possible d'invoquer l'interpréteur Python à partir d'un terminal. Sous Ubuntu Linux (sur les ordinateurs du NO4), il est

possible de faire apparaître une fenêtre terminal via le menu en haut à gauche : *Applications* → *Accessories* → *Terminal*. Il suffit alors de taper la commande `python fichier.py`, où `fichier.py` est le nom de votre fichier source, pour exécuter votre programme.

Exécuter un programme Python via le terminal permet de passer des paramètres au programme, d'une manière similaire au mécanisme de passage de paramètre à une fonction. Le début de votre programme devra contenir la ligne suivante :

```
import sys
```

Lors de l'invocation de l'interpréteur Python via le terminal, des valeurs peuvent être ajoutées après le nom du programme, par exemple :

```
python fichier.py 12 30
```

Ces valeurs peuvent être récupérées par le programme, dans la liste appelée `sys.argv`. Dans l'exemple précédent, l'instruction Python `print sys.argv` affichera les éléments suivants :

```
['fichier.py', '12', '30']
```

Chaque élément de la liste `sys.argv` est de type `str`. Ils peuvent être convertis vers d'autres types à l'aide des fonctions de conversion de types, comme `int()` ou `float()`.

Nous vous demandons de faire en sorte que votre programme soit appelable via la ligne de commande, et gère trois paramètres de la façon décrite ci-dessus. Le premier paramètre sera l'état de départ du tableau, sous forme d'une chaîne de caractères composées de symboles `0` et `1`. Le second paramètre sera le nombre d'étapes successives à afficher. Le troisième paramètre sera l'identifiant de la règle de transition, encodée comme un nombre entier, de la manière décrite dans l'introduction. Le programme affichera alors les états successifs du tableau, sous forme de lignes de `0` et `1`, sans espace. Pour fixer les idées, le second exemple proposé dans l'introduction a été produit avec la commande suivante :

```
python masolution.py 00000010000000000000 20 60
```

Pour en savoir plus. Les automates cellulaires sont à la fois parmi les exemples les plus simples de modèles biologiques et des modèles de calcul élémentaires qui ont très tôt éveillé l'intérêt des chercheurs en informatique théorique. De nombreuses ressources sont accessibles sur internet.

L'Atlas de S. Wolfram, par exemple, contient de nombreux modèles de calculs et d'automates, et est consultable à l'adresse `http://atlas.wolfram.com/`. Des images obtenues à l'aide de la règle 30, avec différentes conditions initiales, sont visibles à la page suivante : `http://atlas.wolfram.com/01/01/30/`. D'autres règles de transition sont décrites à la page suivante : `http://atlas.wolfram.com/01/01/`.

Plus d'informations sur les propriétés des automates cellulaires sont disponibles sur Wikipédia : `http://fr.wikipedia.org/wiki/Automate_cellulaire`.

3 Consignes pour la remise du projet

À respecter scrupuleusement !
Le projet ne sera pas testé à la main, un traitement automatisé sera appliqué. Pour que votre projet soit pris en compte par ce traitement automatisé, il est indispensable que vous respectiez les instructions ci-dessous à la lettre.

LES PROJETS QUI NE RESPECTENT PAS CES INSTRUCTIONS SERONT CONSIDÉRÉS COMME « NON REMIS ».

1. Ce projet est INDIVIDUEL ! Il est bien entendu acceptable d'en discuter entre collègues, mais deux copies trop similaires seront considérées comme frauduleuses ;
2. Toute forme de plagiat sera également sévèrement sanctionnée. A ce sujet, consulter la page des bibiothèques de l'ULB : `http://www.bib.ulb.ac.be/fr/aide/eviter-le-plagiat/`. La notion de plagiat s'applique naturellement aux programmes ou morceaux de programmes.
3. Votre travail devra être remis au plus tard le **le lundi 12 décembre 2011** avant 15 :00 sous deux formats :
   – joint, sous forme d'un fichier projetINFOF206.py, à un **unique** courriel à l'adresse infof206@lit.ulb.ac.be. Le courriel aura pour sujet « Informatique (INFO-F206) : Projet : Remise : <VotreNom> - <VotrePrénom> - <VotreMatricule> », avec les champs <VotreNom>, <VotrePrénom>, <VotreMatricule> remplacés par les informations adéquates,
   – sur papier, au secrétariat étudiants du département d'Informatique (local 2.N8.104 - Plaine), une version imprimée du code source de chacun des fichiers envoyés par courriel. Chaque feuille reprendra le nom et le prénom de l'étudiant, ainsi que son numéro de matricule ;
4. Les dates et heures de remises sont à comprendre au sens strict. Aucun retard ne sera toléré ;
5. Veuillez indiquer sur chaque feuille que vous remettrez les mentions suivantes : **votre nom**, **votre prénom**, **votre matricule** ;
6. Veuillez numéroter vos feuilles clairement et ne remettez pas de feuilles volantes ;
7. Votre projet doit être **dactylographié**. Les projets écrits à la main ne seront **pas corrigés** (0/10) ;
8. Votre code doit être **commenté de manière pertinente**. Veillez par ailleurs à bien soigner sa présentation ;
9. En dehors des cas où l'énoncé le spécifie explicitement, l'utilisation de librairie n'est pas autorisée ;
10. L'ensemble du projet sera à réaliser en Python 2.