Programmation fonctionnelle

Énoncé du Travail

Année 2013-2014

1 L-system

Le système de Lindenmayer (ou L-system) a été inventé en 1968 par le biologiste hongrois Aristid Lindenmayer, afin de modéliser le processus de développement et de prolifération de plantes ou de bactéries.

Son principe de base est relativement simple. Imaginons, par exemple, que nous modélisions des cellules à l'aide de symboles. Nous pouvons alors voir chaque génération obtenue après division des cellules comme étant le résultat du remplacement de chaque symbole de la génération précédente par un ou plusieurs autres symboles.

Formellement, un L-system est décrit par une grammaire composée de 5 éléments :

— Un ensemble fini N de *symboles non terminaux*. Chacun de ces symboles peut éventuellement être accompagné de paramètres. Dans le cadre de ce travail, nous nous limiterons à un paramètre unique prenant une valeur réelle.

— Un ensemble fini Σ de *symboles terminaux*. Chacun de ces symboles peut éventuellement être accompagné de paramètres. Dans le cadre de ce travail, nous nous limiterons à un paramètre unique prenant une valeur réelle.

— Un *axiome* (ou état initial) qui est une séquence finie non vide de symboles de $N \cup \Sigma$.

— Un ensemble de *règles de production*. Chacune de ces règles est de la forme $s \to S$, où s est un symbole de $N \cup \Sigma$ et S une séquence finie de symboles de $N \cup \Sigma$. L'application d'une telle règle à une séquence de symboles y remplace toutes les occurrences de s par la séquence S. Si la séquence S contient des occurrences de s, la règle n'est pas de nouveau appliquée. Chaque règle peut éventuellement être accompagnée d'une probabilité.

  Lorsqu'une règle s'applique à un symbole paramétré, une fonction est associée à celle-ci. Cette fonction exprime comment remplacer le paramètre dans la séquence résultant de l'application de la règle.

  e.g : La règle A[x] $\to$ B[$2x$] A[$0.6x$]
  appliquée au symbole A[3] donnera la séquence B[6] A[1.8].

— Un ensemble de *règles de terminaison*. Chacune de ces règles est de la forme $t \to T$, où t est un symbole de N et T une séquence finie de symboles de Σ. L'application d'une telle règle à une séquence de symboles y remplace toutes les occurrences de t par la séquence T. Si la séquence T contient des occurrences de t, la règle n'est pas de nouveau appliquée.

  Lorsqu'une règle s'applique à un symbole paramétré, une fonction est associée à celle-ci. Cette fonction exprime comment remplacer le paramètre dans la séquence résultant de l'application de la règle.

En pratique, nous utiliserons la grammaire de la manière suivante :

— Nous appelons *génération d'ordre n* d'un L-system le résultat de l'application de n étapes successive de production suivi par une étape de terminaison à l'axiome initial.

— Une étape de production (terminaison) consiste en l'application de l'ensemble des règles de production (terminaison) à une séquence finie de symboles.
  En d'autres termes, pour chaque symbole de la séquence, on applique la règle correspondante si elle existe, et on laisse le symbole inchangé sinon.

  La probabilité éventuelle associée à une règle $s \to S$ détermine la probabilité de remplacer une occurrence de s par la séquence S.
  Une règle à laquelle aucune probabilité n'a été associée aura une probabilité de remplacement égale à 1.
  Remarque : La somme des probabilités associées aux règles s'appliquant au symbole s doit forcément être inférieure ou égale à 1.

1.1 Exemple : Fibonacci

Remarque : cet exemple simple n'utilise pas de paramètres ou de probabilités.

— $N = \{\text{A, B}\}$
— $\Sigma = \{\text{F, G}\}$
— Axiome : A

— Règles de production :

  A $\to$ B
  B $\to$ A B

— Règles de terminaison :

  A $\to$ F
  B $\to$ G

Pour produire une génération d'ordre 5, on part de l'axiome et on dérive de nouvelles séquences de symboles en appliquant 5 étapes de production successives.

On a donc par exemple ici :

0. A
1. B
2. A B
3. B A B
4. A B B A B
5. B A B A B B A B

On applique ensuite une étape de terminaison pour obtenir notre séquence finale correspondant à une génération d'ordre 5 de notre L-system :

G F G F G G F G

1.2 Interprétation graphique

Les séquences finales obtenues à partir de L-systems peuvent être interprétées graphiquement. Dans ce travail, nous nous limiterons a une représentation en deux dimensions, mais une représentation en 3D est envisageable.

Considérons par exemple un L-system dont les symboles terminaux sont
$\Sigma = \{\mathrm{F}, +, -\}$ et auxquels nous associons l'interprétation graphique suivante :

F	Tracer une ligne d'une unité dans la direction courante
$+$	Effectuer une rotation de a degrés dans le sens trigonométrique
$-$	Effectuer une rotation de a degrés dans le sens des aiguilles d'une montre,

a étant un angle donné.

On peut alors associer à la séquence suivante une interprétation graphique :

F + F − − F + F + F + F − − F + F − − F + F − − F + F + F + F − − F + F

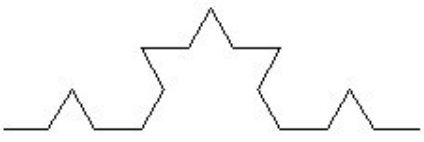

FIGURE 1 – Interprétation graphique (angle=60 °)

Dans ce travail, les symboles terminaux se limiteront à un sous-ensemble de

$$\{\mathrm{F},\ \mathrm{F}[x],\ \mathrm{f},\ \mathrm{f}[x],\ +,\ +[x],\ -,\ -[x],\ <,\ >,\ \mathrm{P}[x],\ *,\ \mathrm{p}\}$$

auxquels nous associons l'interprétation graphique suivante :

F	Tracer une ligne d'une unité dans la direction courante
F[x]	Tracer une ligne de x unités dans la direction courante
f	Se déplacer d'une unité dans la direction courante
f[x]	Se déplacer de x unités dans la direction courante
$+$	Effectuer une rotation de a degrés dans le sens trigonométrique
$+[x]$	Effectuer une rotation de $x * a$ degrés dans le sens trigonométrique
$-$	Effectuer une rotation de a degrés dans le sens des aiguilles d'une montre
$-[x]$	Effectuer une rotation de $x * a$ degrés dans le sens des aiguilles d'une montre
$<$	Sauvegarder la position courante
$>$	Restaurer la position précédemment sauvegardée selon une politique *Last In First Out*
P[x]	Sauvegarder le polygone courant inachevé (si il en existe un) et créer un nouveau polygone courant dont la couleur de remplissage est x.
$*$	Ajouter un point au polygone courant correspondant à la position courante.
p	Dessiner le polygone courant et restaurer comme polygone courant un polygone précédemment sauvegardé (si il en existe) selon une politique *Last In First Out*.

(Dans cette interprétation, a est un angle constant donné).

2 Travail

Il est demandé d'écrire un programme Scheme pouvant prendre en entrée la description d'un L-system, représenté dans le format de votre choix, ainsi qu'un entier n. Le programme devra être capable de produire en sortie l'interprétation graphique (sect.1.2) correspondant à une génération d'ordre n du L-system, exprimée dans le format SVG.

Remarques :

— Ce travail est à réaliser par groupe de deux étudiants.

— Votre travail doit pouvoir fonctionner sous Linux (*DrRacket*).
(Les machines ms800 seront utilisées comme référence.)

— Ce travail doit être envoyé sous la forme d'une archive tar compressée nomée `scheme_nom1_nom2.tar.gz`.

— Votre archive doit contenir un rapport au format `PDF`, le programme Scheme réalisé, un programme permettant d'exécuter au minimum les six exemples donnés dans la section suivante, ainsi que les résultats SVG correspondants.

— Votre rapport expliquera tous vos choix non triviaux ainsi que tout ce vous jugerez utile à la correction.

— Votre travail doit être envoyé avant le 15 mai 2014 à 23h59 à l'adresse `lens@montefiore.ulg.ac.be`. L'email doit être envoyé depuis votre adresse ULg, avec `[INFO-0054 Projet]` comme sujet et avec l'adresse de votre co-auteur en copie.

— Le programme pourra éventuellement prendre d'autres paramètres en entrée, comme le nom du fichier de sortie, un facteur d'échelle pour l'affichage, ...

— N'oubliez pas de tester votre programme avec d'autres exemples !

— Dans un second temps, vous pouvez imaginer, rajouter d'autres types de paramètres (épaisseur, couleur des lignes, 3D, ...), mais ce n'est pas obligatoire pour la réussite de ce travail.

3 Exemples

Remarque : les fichiers SVG correspondant aux exemples sont disponibles sur la page `http://www.montefiore.ulg.ac.be/~lens/scheme/scheme.html`.

3.1 Exemple 1 (Flocon de Koch)

— $N = \{\}$

— $\Sigma = \{\mathrm{F}, +, -\}$

— Axiome : F + + F + + F

— Règles de production :

$\mathrm{F} \rightarrow \mathrm{F} - \mathrm{F} + + \mathrm{F} - \mathrm{F}$

— Règles de terminaison : /

— angle = 60 °

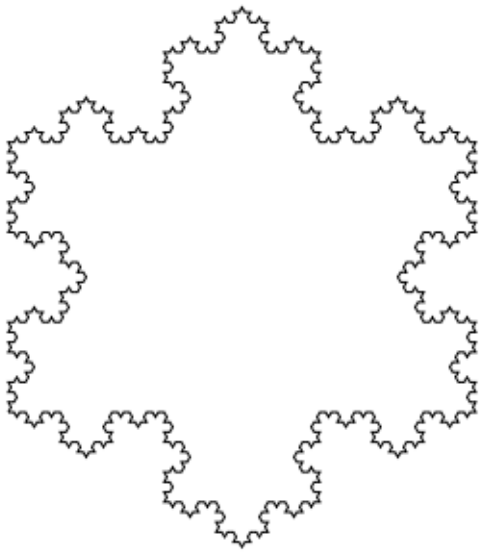

FIGURE 2 – Flocon de Koch ($n = 4$)

3.2 Exemple 2 (Dragon curve)

— $N = \{\mathrm{D}, \mathrm{E}\}$

— $\Sigma = \{\mathrm{F}, +, -\}$

— Axiome : D

— Règles de production :

$\mathrm{D} \to - \ \mathrm{D} \ + \ + \ \mathrm{E}$
$\mathrm{E} \to \mathrm{D} \ - \ - \ \mathrm{E} \ +$

— Règles de terminaison :

$\mathrm{D} \to - \ - \ \mathrm{F} \ + \ + \ \mathrm{F}$
$\mathrm{E} \to \mathrm{F} \ - \ - \ \mathrm{F} \ + \ +$

— angle $= 45^\circ$

FIGURE 3 – Dragon curve ($n = 10$)

3.3 Exemple 3 (Tapis de Sierpinski)

— $N = \{\}$

— $\Sigma = \{\mathrm{F}, +, -, <, >\}$

— Axiome : $\mathrm{F} - \mathrm{F} - \mathrm{F} - \mathrm{F}$

— Règles de production :

$\mathrm{F} \rightarrow \mathrm{F} < - \mathrm{F} - \mathrm{F} > \mathrm{F} < - \mathrm{F} - \mathrm{F} - \mathrm{F} > \mathrm{F}$

— Règles de terminaison : /

— angle = 90°

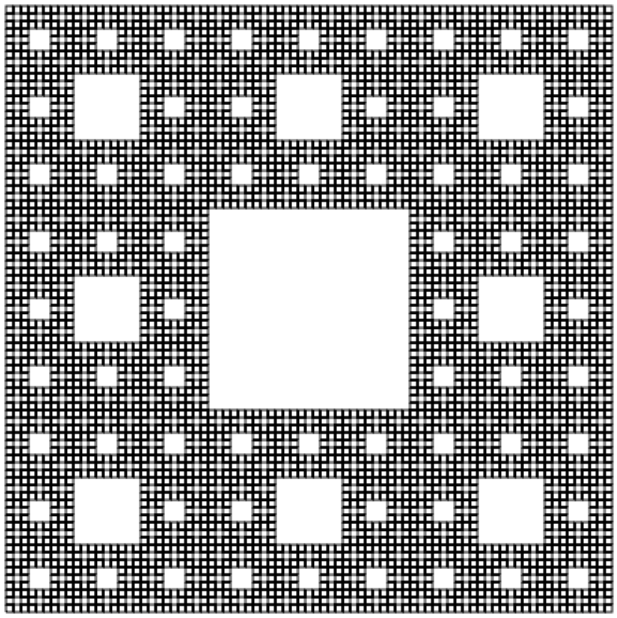

FIGURE 4 – Tapis de Sierpinski ($n = 4$)

3.4 Exemple 4 (Arbre)

— $N = \{\}$

— $\Sigma = \{\mathrm{F}, +, -, <, >\}$

— Axiome : F

— Règles de production :

$\mathrm{F} \rightarrow \mathrm{F} < + \mathrm{F} > \mathrm{F} < - \mathrm{F} > \mathrm{F}$ (prob 0.33)
$\mathrm{F} \rightarrow \mathrm{F} < + \mathrm{F} > \mathrm{F}$ (prob 0.33)
$\mathrm{F} \rightarrow \mathrm{F} < - \mathrm{F} > \mathrm{F}$ (prob 0.34)

— Règles de terminaison : /

— angle $= 25.7^\circ$

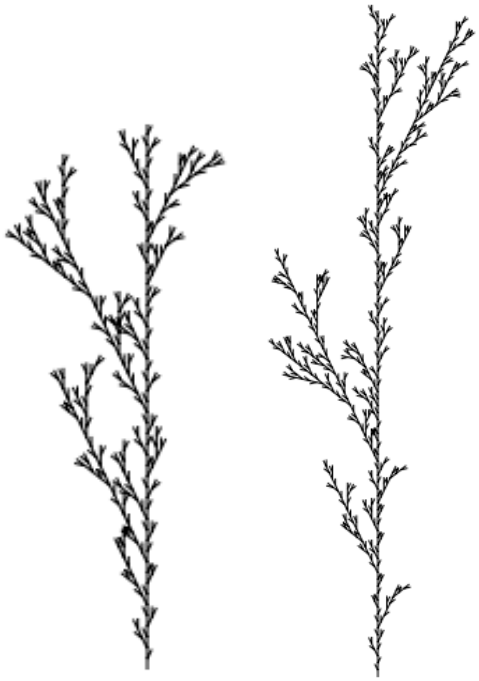

FIGURE 5 – Arbres ($n = 5$)

3.5 Exemple 5 (Plante)

— $N = \{\mathrm{B}[x]\}$

— $\Sigma = \{\mathrm{F}[x], +[x], -[x], <, >\}$

— Axiome : B[1]

— Règles de production :

$$\begin{aligned}\mathrm{B}[x] \rightarrow\ & \mathrm{F}[x] < +[5]\ \mathrm{B}[0.5x] > < -[7]\ \mathrm{B}[0.5x] > -[1]\ \mathrm{F}[x] \\ & < +[4]\ \mathrm{B}[0.5x] > < -[7]\ \mathrm{B}[0.5x] > -[1]\ \mathrm{F}[x] \\ & < +[3]\ \mathrm{B}[0.5x] > < -[5]\ \mathrm{B}[0.5x] > -[1]\ \mathrm{F}[x]\ \mathrm{B}[0.5x]\end{aligned}$$

— Règles de terminaison :

$$\mathrm{B}[x] \rightarrow \mathrm{F}[x]$$

— angle $= 8\,^{\circ}$

FIGURE 6 – Plante ($n = 4$)

3.6 Exemple 6 (Pavage de Penrose)

— $N = \{\text{A, B, C, D}\}$

— $\Sigma = \{\text{F, +, −, P}[x], *, \text{p}\}$

— Axiome : < B > + + < B > + + < B > + + < B > + + < B >

— Règles de production :

A → C + + P[0x45d] ∗ D ∗ − − − − B ∗ < − C ∗ − − − − A p > + +
B → + P[0xfd0] ∗ C ∗ − − D ∗ < − − − A * − − B p > +
C → − P[0xfd0] ∗ A ∗ + + B ∗ < + + + C ∗ + + D p > −
D → − − P[0x45d] ∗ C ∗ + + + + A ∗ < + D ∗ + + + + B p > − − B

— Règles de terminaison :

A → F
B → F
C → F
D → F

— angle = 36 °

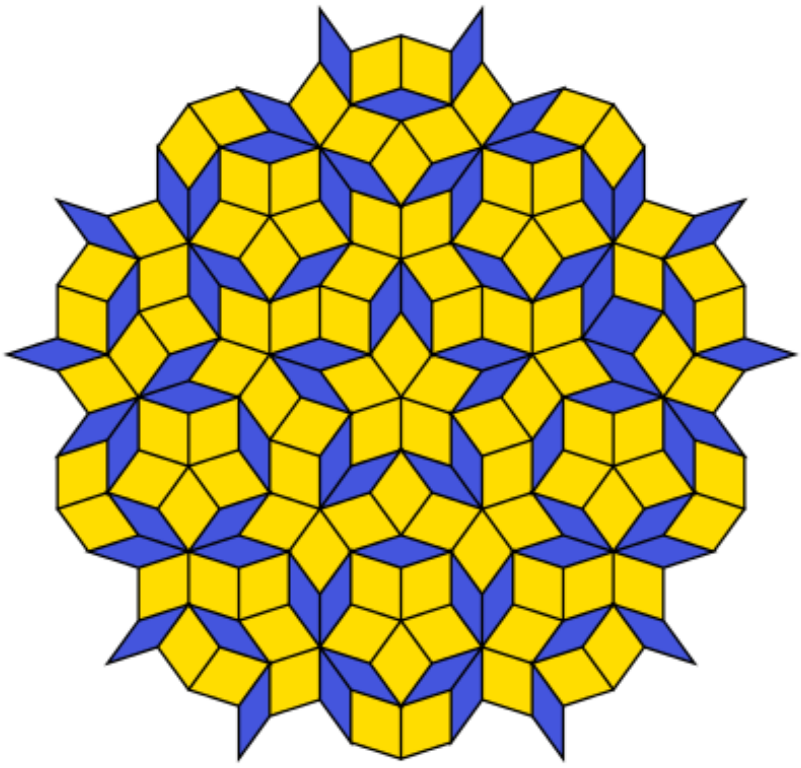

FIGURE 7 – Pavage de Penrose ($n = 4$)

4 Annexes

4.1 Scheme

4.1.1 Écriture dans un fichier

Les mécanismes permettant d'écrire dans un fichier à partir d'un programme Scheme sont décrits sur la page

`http://docs.racket-lang.org/reference/file-ports.html`

(On s'intéressera en particulier à la fonction `with-output-to-file`.)

4.1.2 Random

Les fonctions permettant de générer un nombre aléatoire dans un programme Scheme sont décrits sur la page

`http://docs.racket-lang.org/reference/numbers.html`

(On s'intéressera en particulier à la fonction `random`.)

4.1.3 Fichiers sources multiples

Pour plus de lisibilité, il est conseillé de répartir les fonctions composant le programme dans plusieurs fichiers source.

Pour ce faire, on utilisera la fonction `(load "file.scm")` pour charger un fichier dans un autre.

4.2 SVG

Une spécification du format SVG peut être consultée à l'adresse :

`http://www.w3.org/Graphics/SVG/`

Remarques :

— L'élément `path` du format est, par exemple, particulièrement bien adapté à notre problème.

— Il peut être intéressant d'utiliser la balise `viewBox` pour sélectionner la zone d'intérêt dans l'image SVG.

— N'hésitez pas à consulter les exemples fournis sur la page :
`http://www.montefiore.ulg.ac.be/~lens/scheme/scheme.html`