Chapitre

1

MORPHOGÉNÈSE INFORMATIQUE

1. Comprendre le développement pour expliquer le vivant

1.1. L'animal-machine

En 1739, Jacques de Vaucanson (1709 – 1782) présente à l'Académie des Sciences un automate fameux, le « Canard Digérateur », un chef-d'œuvre de simulation anatomique comptant plus de 400 pièces en mouvement reproduisant les principales fonctions vitales (respiration, digestion, locomotion) : l'animal battait des ailes, mangeait du grain et déféquait (les grains étaient digérés par dissolution).

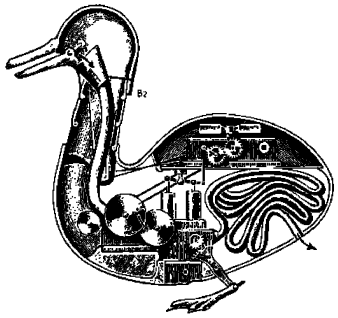

Figure 1.1. Le canard de Vaucanson. Voltaire décrira Vaucanson par ces vers : « Le hardi Vaucanson, rival de Prométhée, Semblait, de la nature imitant les ressorts, Prendre le feu des cieux pour animer les corps. ».

En réalisant ces « anatomies mouvantes », Jacques de Vaucanson est très certainement inspiré par la philosophie bio-mécaniste de René Descartes (1596 – 1650) qui réduit les organes du corps humain aux pièces d'une machine « agencée par

Dieu ». Descartes pense en effet comprendre la Vie en comparant le corps à une machine : on peut expliquer les principales fonctions corporelles – la digestion, la locomotion, la respiration, mais aussi la mémoire et l'imagination – comme si elles résultaient d'un automate, à l'image d'une horloge destinée à montrer les heures par la seule disposition de ses roues et contrepoids.

Mais la Reine Christine de Suède, que René Descartes essaya de convaincre, lui rétorqua qu'elle y croirait quand il lui aurait montré comment une horloge peut se reproduire...

La démonstration prit trois siècles : si un siècle plus tard les automates de Vaucanson imitaient les grandes fonctions physiologiques, ils ne pouvaient toujours pas se reproduire et il fallu attendre un article de John Von Neumann en 1951, *The General and Logical Theory of Automata*, pour se convaincre qu'une machine peut effectivement construire une copie d'elle-même.

Pour répondre à la question de la Reine Christine, il a fallu définir précisément ce qu'on entendait par « machine » et ce qu'on entendait par « reproduction ». Pour Von Neumann, qui a sur ce point une approche très fonctionnaliste, la mécanique est *in fine* ce qui peut être réduit à un programme d'ordinateur et la reproduction correspond à dupliquer ce programme. Attention, il ne s'agit pas de recopier un fichier contenant un programme par une commande du système d'exploitation de l'ordinateur, mais de faire en sorte que le fonctionnement du programme produise une description complète et fonctionnelle du programme lui-même. La figure 1.2 donne l'exemple d'un tel programme écrit dans le langage `C` : son exécution produit un fichier qui contient l'exacte copie de son propre code. On parle de code *auto-reproducteur*.

```
#include<stdio.h>

main(){char*c="\\\"#include<stdio.h>%cmain(){char*c=%c%c%c%.
102s%cn%c ;printf(c+2,c[102],c[1],*c,*c,c,*c,c[1]) ;exit(0) ;}
\n" ;printf(c+2,c[102],c[1],*c,*c,c,*c,c[1]) ;exit(0) ;}
```

Figure 1.2. Un code auto-reproducteur. Ce programme est constitué de deux lignes de code dans le langage de programmation `C`. La 2ème ligne du programme (qui commence par `main`) a été arbitrairement typographiée sur trois lignes pour des raisons de lisibilité.

L'objectif de Von Neumann était clairement de montrer que les processus du vivant sont réductibles à des processus mécaniques, décrits par des opérations pouvant être exécutées de manière autonome, sans l'aide d'un « cornac caché » : une machine. Et pour Von Neumann, comme pour la Reine Christine, la reproduction et le développement sont une caractéristique spécifique du vivant. Mais pour Von Neumann cette caractéristique est juste une propriété particulière possédée par certaines machines, et non pas une qualité transcendant les processus physiques pour donner un statut particulier aux processus biologiques. L'existence d'une machine, d'un automate, capable de reproduction, est donc un élément de réponse crucial dans un débat fort ancien opposant le statut de la biologie à celui de la physique.

Ce débat n'était pas facile à trancher, la reproduction étant l'un des processus les plus fondamentaux de la vie des organismes et semblant réfractaire à toute explication physique. En effet, l'intuition nous suggère que si une machine A peut produire comme résultat de son fonctionnement une machine B, alors A doit contenir d'une manière ou d'une autre la description complète de B ainsi que celle des

mécanismes spécifiques lui indiquant comment utiliser cette description pour produire effectivement (construire) B. Cette description doit être interne à A sinon on aurait affaire à un mécanisme de copie et non de reproduction. On devrait donc pouvoir définir une certaine mesure de complexité et montrer que la complexité de A est nécessairement plus grande que celle de B. Notre intuition est ici erronée.

1.2. De l'auto-reproduction au développement

Mais si les biologistes contemporains se posent les mêmes questions que les philosophes et les reines des siècles passés, la compréhension des mécanismes de la reproduction passe aujourd'hui par l'élucidation des *processus* qui conduisent de la cellule germinale à l'organisme complet : il s'agit de comprendre étape par étape la *construction au cours du temps* d'un organisme complet à partir des très nombreuses interactions locales des éléments constituant l'organisme. On parle alors de *développement*.

Les éléments de réponse apportés par Von Neumann sont très abstraits : ils reposent sur la description d'un automate cellulaire qui reproduit au cours du temps la configuration d'un sous-domaine spatial dans une région voisine. Un automate cellulaire peut se décrire par un réseau *prédéfini* de sites, appelés encore cellules, chaque cellule possédant un état pris dans un ensemble fini. L'état de chaque cellule est mis à jour suivant une règle d'évolution prédéfinie qui combine l'état à l'instant t de la cellule et de ses voisines pour calculer l'état à l'instant $t+1$. Le fonctionnement de l'automate correspond à la mise à jour, dans une suite d'étapes discrètes, de l'état des cellules (cf. figure 1.3).

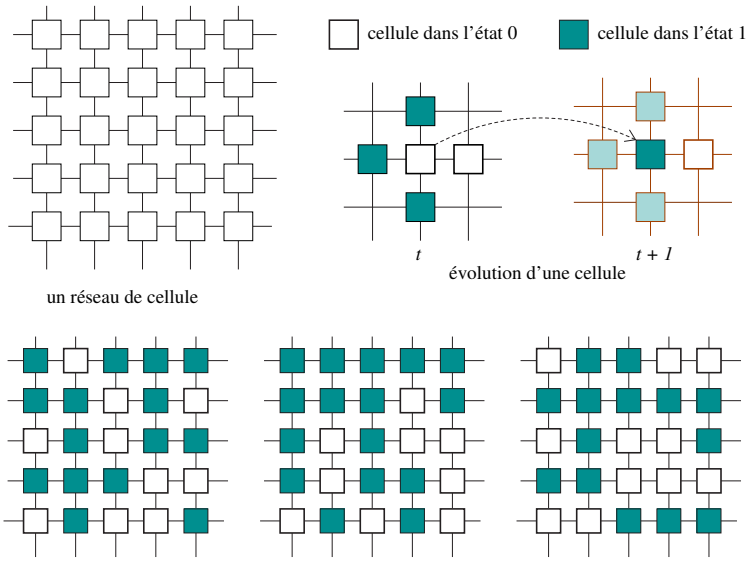

Figure 1.3. Un automate cellulaire est un réseau de cellules (les cellules voisines sont reliées par des liens et ici le réseau est une grille rectangulaire). Chaque cellule possède un état (ici 0 ou bien 1). La règle d'évolution utilisée ici est : l'état d'une cellule est la somme modulo 2 de l'état des cellules voisines. Un exemple d'évolution d'une seule cellule est donnée en haut à droite. Les trois réseaux au-dessous montrent 3 étapes successives de l'évolution de l'automate. La règle est appliquée simultanément à toutes les cellules. L'automate auto-reproducteur de Von Neumann est un modèle de ce type, où les règles d'évolution induisent la reproduction de la configuration initiale d'une région donnée du réseau dans une région adjacente.

On voit que l'on est très loin des mécanismes moléculaires auxquels les biologistes contemporains veulent réduire les phénomènes biologiques. L'existence d'un automate auto-reproducteur est un argument qui indique qu'il n'y a pas de problème de principe à l'existence d'une telle machine, mais qui ne nous éclaire pas sur le « comment » mis en œuvre dans les processus biologiques. Cependant, les notions de programme, de code, d'automate, de mémoire, d'information, ont envahi la biologie et prennent une valeur explicative, en particulier en biologie du développement [5] : les biologistes ont en effet besoin de modèles et de métaphores pour comprendre (i.e., représenter, analyser et interpréter) l'immense masse de leurs données expérimentales. Par exemple, la notion de *code génétique* joue pour la cellule vivante un rôle similaire à la règle qui spécifie l'évolution de l'état d'un site dans l'automate de Von Neumann.

1.3. Le développement comme un système dynamique

La notion de *système dynamique* permet de formaliser la notion de processus de développement. Un système dynamique (SD) est caractérisé par des observations qui évoluent dans le temps. Ces observations sont les *variables* du système et sont reliées par certaines relations. Ces variables rendent compte des propriétés pertinentes du système (biologiques, physiques, chimiques, sociologiques, ...). À un instant donné, elles prennent une valeur et l'ensemble de ces valeurs constitue l'*état du système*. L'ensemble de tous les états possibles d'un système constitue son *espace d'état* (ou espace des configurations).

Par exemple, une pierre qui tombe est un système caractérisé par les variables *position* et *vitesse* de la pierre. Ces deux variables ne sont pas indépendantes : si on conçoit la position de la pierre comme une fonction du temps, la vitesse est la dérivée de cette fonction.

La séquence temporelle des états du système est appelée une *trajectoire*. Un SD est un moyen formel pour spécifier comment on passe d'un point dans l'espace des configurations (un état) à un autre (l'état suivant). Ceci peut se faire directement par une fonction (la fonction d'évolution du système) ou indirectement en donnant des contraintes (équations) sur l'état futur possible (qui n'est pas nécessairement unique si le système n'est pas déterministe). Divers formalismes mathématiques correspondent à cette notion très générale de système dynamique. Par exemple les variables peuvent prendre des valeurs continues ou discrètes. De la même manière, l'avancement du temps peut progresser par pas discrets ou bien se faire continuement. Des exemples de formalismes correspondant à ces cas sont listés dans la table 1.1.

Tableau 1.1. Trois exemples de formalisme utilisés pour spécifier un système dynamique suivant la nature discrète ou continue des variables et du temps. Les itérations de fonctions correspondent aux suites $x_{n+1} = f^{n+1}(x_0) = f(x_n)$ pour une certaines fonctions f sur $\mathbb{R}$. De nombreux autres formalismes ont été étudiés.

C : continu D : discret	équation différentielle	itérations de fonctions	automate fini
Temps	C	D	D
Etat	C	C	D

Dans les cas simples, la trajectoire d'un système dynamique peut s'exprimer explicitement par une fonction analytique du temps t. Par exemple, dans le cas de la pierre qui tombe, les équations différentielles $dx/dt = v$ et $dv/dt = \mathbf{g}$ peuvent s'intégrer explicitement pour donner la distance parcourue par la pierre en fonction du temps : $x = \mathbf{g}t^2/2$.

Dans les cas un peu plus complexes, une formule analytique donnant la trajectoire n'existe pas et la simulation par ordinateur est alors une approche privilégiée pour étudier les trajectoires du système. Par ailleurs, on peut s'intéresser non pas à une trajectoire particulière mais aux propriétés qualitatives vérifiées par toutes les trajectoires possibles, comme par exemple : « si on attend assez longtemps, le système finira par prendre un état bien déterminé qu'il ne quittera plus » ou encore « si on passe par ces états, on n'y repassera jamais ». On parle de *propriétés émergentes* quand il n'existe aucun moyen plus rapide pour les prédire que de les observer ou de les simuler. Notons que des SD dont la spécification est très simple peuvent donner des trajectoires extrêment complexes (on parle parfois de *comportement chaotique*) ; d'autre part, le calcul de la trajectoire du système peut être coûteux en temps d'ordinateur et exiger beaucoup de mémoire.

■ La structure des états

Une autre caractéristique importante permettant de classer les systèmes dynamiques est la structure des états. Dans l'exemple de la pierre qui tombe, la structure d'un état est simple : c'est un couple de vecteurs (vitesse, position).

Très souvent, la structure d'un état reflète l'organisation spatiale du système. Prenons par exemple la diffusion de la chaleur dans un volume. La distribution de la température possède une structure reliée à l'organisation spatiale du volume. On définit alors un champ scalaire associant à chaque point sa température. L'évolution de ce champ suit une loi de diffusion spécifiée par une équation aux dérivées partielles. Celle-ci relie la température en un point p du volume à l'instant $t + dt$ aux valeurs du champ de température en p et dans son voisinage à l'instant t.

Très souvent, seuls des sous-systèmes connectés ou physiquement proches interagissent : on parle de propriété de *localité* (pas d'action à distance). La structure d'un état reflète alors ce découpage en sous-systèmes et la fonction d'évolution respecte cette localité. Pour l'évolution de la température dans un volume, un état associe une température à chaque point du volume V et l'espace des états est donc l'ensemble des fonctions de V dans $\mathbb{R}$. L'équation de diffusion de la chaleur qui gouverne l'évolution du système indique que la température d'un point de V ne dépend que de la température des points voisins.

■ Le développement comme trajectoire d'un système dynamique

Nous avons indiqué plus haut que la notion de *code génétique* s'apparente à la règle qui spécifie l'évolution de l'état d'une cellule dans l'automate de Von Neumann. Dans cette vision, l'évolution complète de l'organisme est codée entièrement dans son matériel génétique ce qui correspond à une approche « tout génétique » où chaque caractère est uniquement déterminé par les gènes. Ce point de vue est largement remis en cause [1] au profit d'une approche plus souple qui permet de réconcilier les points de vue génétique et épigénétique sur le développement. Si les systèmes vivants sont des systèmes dynamiques, ce sont des systèmes ouverts qui interagissent avec leur environnement. Le développement est alors vu comme une

co-construction qui dépend des interactions internes au système aussi bien qu'externe (avec l'environnement). Le matériel génétique ne constitue pas une description complète et suffisante d'un organisme particulier même s'il est indispensable. Par exemple, comme le montre expérimentalement le clonage où l'on introduit un noyau (i.e. le matériel génétique d'une cellule) dans une cellule germinale, le rôle de la machinerie cellulaire est lui aussi primordial.

Cependant, une deuxième propriété caractérise les processus de morphogénèse impliquant le mouvement et la réorganisation de matière : l'espace des états et sa topologie peuvent évoluer avec le temps.

Illustrons cette notion en l'opposant à l'exemple de la pierre qui tombe : la vitesse et la position de la pierre changent à chaque instant mais le système est toujours adéquatement décrit par un couple de vecteurs. Nous dirons dans ce cas que *la structure du SD est statique*. Il en va de même pour l'évolution de la température dans le volume V : V est fixé à l'avance et un état est toujours un élément de $V \to \mathbb{R}$. Dans ces deux exemples, l'espace des états peut être décrit adéquatement à l'origine des temps, avant la simulation ; il correspond à l'espace des mesures du système. La valeur de ces mesures change avec le temps, mais la donnée de l'espace des états et sa topologie ne font pas partie des variables du système et elles ne peuvent pas évoluer au cours du temps.

Le cas est tout autre pour les processus de développement : les processus biologiques constituent des systèmes dynamiques hautement structurés et hiérarchiquement organisés, dont la structure spatiale varie au cours du temps et doit être calculée conjointement avec l'état du système. Nous appellerons ce type de système un *système dynamique à structure dynamique* ou $(\mathbf{SD})^2$ en abrégé.

Le fait que la structure même d'un système biologique soit dynamique a été plusieurs fois souligné sous des noms variés ; on peut citer dans des domaines différents : la notion d'*hyper-cycle* introduite par Eigen et Schuster dans l'étude des réseaux auto-catalytiques [2], la théorie des *systèmes autopoïètiques* formulée par Maturana et Varela [12], les *systèmes à structure variable* développés en théorie du contrôle par Itkis [8], ou encore le concept d'*organisation* introduit par Fontana et Buss pour formaliser et étudier l'émergence de structures fonctionnelles automaintenues dans un ensemble de réactions chimiques [3]. Ces travaux ont tous pour objectif d'appréhender et formaliser l'idée de changement dans la structure d'un système, changement couplé avec l'évolution de l'état du système.

Les $(\text{SD})^2$ sont courants dans les modèles de croissance des plantes et plus généralement en biologie du développement, dans les modèles cellulaires intégrant plusieurs échelles, dans les mécanismes de transport de protéines et de compartimentalisation, etc. Mais on peut citer d'autres domaines où ils sont pertinents comme la modélisation des réseaux mobiles, d'Internet et du Web, le développement des villes, les embouteillages, les processus d'auto-assemblage, les réseaux auto-catalytiques en chimie, les réseaux sémantiques au cours d'un apprentissage, les comportements sociaux, etc.

■ Un exemple

Pour illustrer la notion de $(\text{SD})^2$, prenons pour exemple le développement d'un embryon. L'état de l'embryon est initialement décrit par l'état $s_0 \in \mathcal{S}$ de la cellule germinale (aussi compliquée cette description puisse-t-elle être). Après la première division, il faut décrire l'état de 2 cellules et donc $s_1 \in \mathcal{S} \times \mathcal{S}$. Mais dès que le

nombre n de cellules de l'embryon est assez grand, l'état du système n'est pas décrit adéquatement par un élément de $\mathcal{S}^n$. En effet, cet ensemble décrit uniquement l'état de chaque cellule mais il ne contient pas l'information spatiale permettant de décrire le réseau des cellules (leur positionnement les unes par rapport aux autres). Or ce réseau est de la première importance car il conditionne la diffusion des signaux (chimiques, mécaniques, électriques) entre cellules, et donc *in fine* leur mode de fonctionnement. À chaque mouvement, division ou mort cellulaire, la topologie de ce réseau se transforme. Par exemple, lors de la phase de *gastrulation*, des cellules initialement éloignées deviennent voisines, permettant leur interaction et la modification de leur destin (différenciation des cellules).

1.4. Quel formalisme pour les systèmes dynamiques à structure dynamique ?

Les systèmes dynamiques à structure dynamique sont difficiles à étudier car ils sont difficiles à formaliser. Pour voir pourquoi, reprenons l'exemple du développement d'un embryon.

Nous avons indiqué que la position de chaque cellule changeait au cours du temps ce qui rend difficile par exemple la spécification des processus de diffusion entre les cellules. Une solution qui vient immédiatement à l'esprit est donc de compléter l'état d'une cellule par une information de position et de considérer $\mathcal{T} = \mathcal{S} \times \mathbb{R}^3$ comme une brique de base[1] permettant de construire l'ensemble :

$$\begin{aligned} \mathcal{T}^* &= \mathcal{T} \cup \mathcal{T}^2 \cup \cdots \cup \mathcal{T}^n \cup \ldots \\ &= \mathcal{T} \cup \mathcal{T} \times \mathcal{T}^* \end{aligned}$$

Il est sans doute possible de caractériser un embryon comme un point dans cet espace des phases, mais cela n'apporte pas grand chose : $\mathcal{T}^*$ a en effet très peu de structure intrinsèque et ne nous apprend que peu de choses sur les trajectoires possibles du système. Par exemple, la fonction d'évolution sera très difficile à définir et elle a peu de chances d'être continue.

■ Le problème de la localité

Pour comprendre pourquoi la fonction d'évolution sera difficile à définir, il faut voir que la spécification de la position de chaque cellule par ses coordonnées dans $\mathbb{R}^3$ présuppose la définition d'un repère global. Lors de l'évolution de l'embryon, la croissance d'une cellule va repousser les cellules voisines et de proche en proche, affecter la position de chaque cellule. Entre deux états successifs, il faut donc exprimer le changement de position de chaque cellule par une *transformation globale* des coordonnées. Parce qu'elle doit exprimer globalement les évolutions de chaque position, et que ces évolutions sont dues à de multiples transformations locales concurrentes, l'expression de cette transformation peut être arbitrairement complexe.

L'origine de ce problème est dans l'expression *extrinsèque et globale* de la forme du système[2] et une solution est donc de spécifier de manière intrinsèque la position

[1] Pour simplifier, on ne prend en compte que la position dans $\mathbb{R}^3$ de chaque cellule, mais il faudrait en plus spécifier sa forme qui conditionne son voisinage et les échanges avec les autres cellules (penser à l'aire de la surface de contact entre deux cellules voisines qui conditionne les flux inter-membranaires).

[2] Dans l'approche évoquée, la spécification de la position des cellules se fait dans un repère global indépendant de l'embryon en croissance. Ce repère correspond au repérage des points de l'espace

de chaque cellule, par exemple en complétant l'état $s \in \mathcal{S}$ de chaque cellule par sa distance aux cellules voisines. Dans ce cas, la spécification des changements de position d'une cellule est locale mais, le voisinage de chaque cellule se modifiant, on retrouve le problème d'un espace d'état qui évolue au cours du temps.

■ Le problème de la continuité

Si on reprend l'exemple de la pierre qui tombe, la position de la pierre varie continûment et il en va de même de sa vitesse. L'état du système varie donc continûment au cours du temps et la trajectoire du système est une *fonction continue* du temps dans l'espace des états. Cette continuité permet de raisonner sur les évolutions infinitésimales du système et de poser une équation différentielle qui caractérise la trajectoire. Dans les cas plus compliqués, on obtient une équation aux dérivées partielles (quand l'état possède une structure spatiale) ou un ensemble de telles équations quand il faut prendre en compte plusieurs modes de fonctionnement (en nombre fini et généralement petit).

Dans le cas du développement de l'embryon, ce n'est plus possible : tant qu'il n'y a pas de mouvement[3], de division ou de mort cellulaire, l'état s appartient à un certain $\mathcal{T}^n$ et cette évolution est continue (en supposant que les potentiels électriques, les concentrations chimiques, les contraintes mécaniques, ..., évoluent continûment). Mais les événements morphogénétiques essentiels (par exemple une division cellulaire qui fait passer de $\mathcal{T}^n$ à $\mathcal{T}^{n+1}$) sont discontinus par nature[4].

■ Vers d'autres solutions

La modélisation et la simulation de l'évolution d'un (SD)^2 sont donc particulièrement ardues car il est difficile de définir à la fois la structure et la dynamique du système, l'une dépendant de l'autre. L'exemple précédent souligne l'inadéquation des formalismes globaux et des formalismes continus (on veut exprimer une évolution comme une succession d'événements morphogénétiques discrets qui correspondent à des ruptures et des changements qualitatifs). Cependant, la description de ces systèmes reste possible et les lois d'évolution sont alors souvent informellement décrites comme un ensemble de transformations *locales* agissant sur un ensemble organisé d'entités discrètes.

Face à ces difficultés, plusieurs chercheurs ont proposé d'utiliser les *systèmes de réécriture* pour formaliser ce type de description.

---

dans lequel la forme est plongée, et non à un processus intrinsèque à la forme en croissance : les lois gouvernant le mouvement, la division et la mort cellulaire seraient les mêmes si l'embryon se développait dans un volume torique (mais le résultat pourrait être différent car les voisinages des cellules seraient alors différents).

[3] Les mouvements cellulaires suffisent à changer la topologie et donc l'interaction des cellules.

[4] Dans l'exemple qui nous sert de fil conducteur, les événements morphogénétiques sont discontinus parce que la modélisation est faite au niveau cellulaire. On aurait pu modéliser la concentration des différentes molécules en chaque point de l'espace, ce qui éviterait peut-être ce problème de discontinuité (le mouvement de chaque molécule étant *a priori* continu). Mais un autre problème surgit alors : comment retrouver dans ces concentrations les entités biologiques qui nous intéressent : cellules, tissus, organes, etc?

2. Les systèmes de réécritures

2.1. Introduction

Les systèmes de réécriture (SR) font partie des formalismes que les informaticiens se sont appropriés et ont développés en particulier pour modéliser les changements d'état d'un processus. Un système de réécriture est un mécanisme permettant de définir le remplacement d'une sous-partie d'un objet par un autre. Habituellement les objets concernés sont des *termes* qu'on peut représenter par des arbres informatiques dont les sommets intérieurs sont des opérations et les feuilles des constantes (cf. figure 1.4). Un SR est défini par un ensemble de règles et une règle est un couple noté $\alpha \to \beta$. Une règle $\alpha \to \beta$ indique comment un sous-terme α peut se remplacer par un terme β.

■ Un exemple

Prenons les expressions arithmétiques et la règle $0 + x \to x$. Intuitivement cette règle spécifie que toute expression pouvant se mettre sous la forme « 0 ajouté à quelque chose dénoté par x » peut se réécrire plus simplement comme « la chose dénotée par x ». Ainsi l'expression $e = 1 + (0 + 3)$ peut se réécrire en $e' = 1 + 3$ en appliquant la règle précédente au sous-terme $(0 + 3)$ de e. Nous écrirons aussi $e \to e'$ pour indiquer que e peut se réécrire en e' par une seule application de règle. La séquence $e \to e_1 \to \cdots \to e_n \to e'$ est appelée une *dérivation* de e. Nous dirons que e est une *forme normale* s'il n'existe pas de e' tel que $e \to e'$.

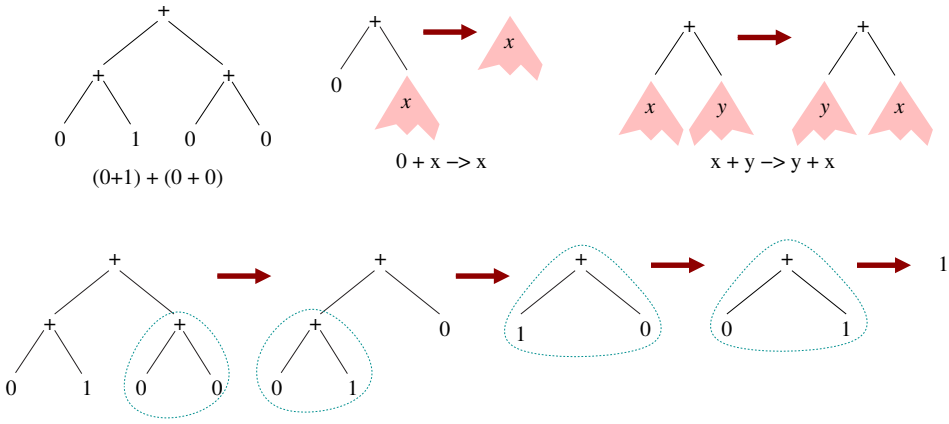

Figure 1.4. Représentation du terme $(0 + 1) + (0 + 0)$ et application des deux règles $0 + x \to x$, $x + y \to y + x$. À chaque réduction, la stratégie est ici d'appliquer à chaque fois une seule règle. Le sous-arbre filtré par la partie gauche de la règle qui va être appliquée est entouré d'un tireté. Les applications sont non déterministes dans le sens où on aurait pu choisir d'autres applications à chaque étape. Par exemple, pour la première réduction on aurait pu appliquer la même règle $0 + x \to x$ au sous-arbre gauche de la racine plutôt qu'au sous-arbre droit. On aurait pu aussi choisir d'appliquer la règle $x + y \to y + x$ à n'importe quel sommet intérieur (3 possibilités). Le terme final obtenu est la constante 1 et c'est une *forme normale* pour les deux règles.

■ **SR et procédure de décision dans une théorie équationnelle**

La motivation originale des SR est de fournir une procédure de décision dans les théories équationnelles. Dans ces théories, on cherche à prouver de manière automatique l'égalité de deux termes complexes en utilisant uniquement des égalités élémentaires prédéfinies. L'idée est d'orienter les équations (par exemple d'orienter l'égalité $0 + x = x$ en une règle $0 + x \to x$) et d'utiliser les règles obtenues pour dériver la forme normale e' d'un terme e. La forme normale e' est équivalente à e (puisque chaque substitution transforme un sous-terme en un terme équivalent) et peut s'interpréter comme une simplification de e. Deux termes e_1 et e_2 sont alors équivalents dans la théorie s'ils se dérivent dans la même forme normale e. Par exemple e_1 défini par $0 + (1 + 3)$ est équivalent à e_2 défini par $1 + (0 + 3)$ car e_1 et e_2 se dérivent dans la même forme normale $e : 1 + 3$.

Pour que cette procédure de décision aboutisse toujours, il faut qu'il existe une forme normale pour chaque expression (propriété de *normalisation*) et que chaque expression possède une seule forme normale (propriété de *confluence*). Ces deux propriétés ne suffisent pas tout à fait à rendre cette procédure de décision calculable automatiquement ; il faut aussi à chaque étape décider d'une dérivation, c'est-à-dire quel sous-terme doit être réécrit et par quelle règle : c'est la *stratégie* d'application des règles.

La théorie des SR, voir [15, 16], est développée principalement en algèbre et en logique mais s'applique à presque toutes les branches de l'informatique (des réseaux de Pétri au calcul symbolique, de la théorie de la démonstration au lambda-calcul). Un résultat clé est que les SR, vus comme des processus de calcul, sont *Turing-complets* (tout processus calculable, i.e. décrit par une machine de Turing, peut se formaliser par un SR). L'utilisation de règles pour transformer un terme est une opération si fondamentale que plusieurs environnements génériques ont été développés pour définir et appliquer des SR (voir, entre autres, les sites des projets ELAN [18] et MAUDE [17]). Les outils se différencient par les termes pris en compte, les motifs α permis en partie gauche d'une règle pour sélectionner un sous-terme, et par les stratégies d'application qu'on peut définir.

2.2. Système de réécriture et simulation des systèmes dynamiques

La présentation précédente suggère qu'une règle $\alpha \to \beta$ spécifie un terme β équivalent au terme α (et plus simple). Mais on peut interpréter cette règle comme l'évaluation d'un calcul (l'expression β est le résultat du calcul de l'expression α) ou encore comme l'évolution d'un sous-système passant de l'état α à l'état β. Les SR permettent donc de modéliser et de simuler des SD :

- un état est représenté par un terme et un sous-terme représente l'état d'un sous-système ;
- la fonction d'évolution est codée par les règles du SR de la manière suivante : la partie gauche de la règle correspond à un sous-système dont les éléments sont en *interaction*, et la partie droite de la règle correspond au résultat de cette interaction.

Ainsi, la dérivation d'un terme s correspond à une trajectoire possible d'un SD partant de l'état initial s. Une règle de réécriture correspond alors à la spécification de l'évolution d'un sous-système. Une forme normale correspond à un point fixe de la trajectoire (le système est à l'équilibre et aucune évolution ne peut se produire).

■ **Exemple.**

Pour le développement de l'embryon, une règle $c \oplus i \to c'$ peut s'interpréter comme une cellule dans l'état c qui, recevant un signal i, évolue dans l'état c' ; une règle $c \to c' \oplus c''$ représente une division cellulaire ; une règle $c \to \emptyset$ (c donne rien) représente une apoptose ; etc [29, 20]. L'idée est que l'évolution d'un biosystème est spécifiée par des règles de réécriture dont la partie gauche sélectionne une entité du système et les messages qui lui sont adressés, et la partie droite décrit le nouvel état de l'entité. L'opérateur $\oplus$ qui apparaît dans la règle dénote la composition des entités locales en un système global (dans notre exemple, l'agrégation des cellules en un embryon). La capacité à représenter dans le même formalisme les changements d'état et l'apparition ou la disparition de cellules font des SR un bon candidat pour la modélisation des $(SD)^2$.

2.2.1. La gestion du temps

Un point important dans la modélisation d'un $(SD)^2$ est le traitement du temps. Le modèle de temps favorisé dans le cadre des SR est clairement un modèle événementiel, atomique et discret : le temps passe quand une évolution se produit quelque part dans le système, l'application d'une règle correspond à un événement et spécifie un changement atomique et instantané dans l'état du système. La notion de durée n'est pas prise en compte (bien qu'elle puisse être prise en compte dans ce formalisme en considérant des événements de début et de fin). Le choix d'une stratégie d'application permet un certain contrôle sur le modèle du temps : par exemple une application parallèle maximale des règles pour passer d'un état global à un autre correspond à une dynamique synchrone, alors que l'application d'une seule règle correspond à une dynamique asynchrone.

2.2.2. La gestion de l'espace

Une règle de la forme $c \oplus i \to c'$ suppose qu'un signal i produit par une certaine cellule va atteindre sa cible c ailleurs dans le système. L'opération $\oplus$ utilisée pour amalgamer les états des sous-systèmes et les messages d'interaction en l'état d'un système complet doit donc exprimer les dépendances spatiales et l'organisation fonctionnelle du système étudié.

La notion de réécriture a été principalement développée et étudiée pour la réécriture de termes. Ceux-ci représentent une restriction sévère des SR car leur utilisation implique de coder la structure hautement organisée des $(SD)^2$ par un arbre. De ce codage dépend la possibilité de définir des règles d'évolution. C'est un travail qui demande beaucoup de créativité et d'intuition. Il est difficile de représenter de manière satisfaisante l'organisation en molécules, compartiments, cellules, tissus, organes, individus d'un système biologique, ce qui motive l'extension de la notion de réécriture à des structures plus sophistiquées que les termes (par exemple on peut définir une notion de réécriture sur un graphe, voir aussi [19, 21]).

Cependant, même en se restreignant à des arbres, les SR proposent des exemples remarquables de modélisation de $(SD)^2$, en particulier dans le domaine biologique. En effet, en jouant sur les propriétés des opérateurs, il est possible de modéliser plusieurs types d'organisation. Dans les sections suivantes, nous donnons des exemples où :

– l'opération $\oplus$ est associative et commutative, ce qui permet de modéliser une « soupe chimique » ;
– plusieurs opérations peuvent être considérées simultanément afin d'introduire l'idée de compartimentalisation ;
– l'opération $\oplus$ est simplement associative, ce qui permet de représenter des séquences et des arborescences.

3. Réécriture de multi-ensembles et modélisation chimique

L'état d'une solution chimique peut se représenter par un *multi-ensemble* : un multi-ensemble est un ensemble où un élément peut apparaître plusieurs fois, à l'image d'une solution chimique chimique où plusieurs molécules de la même espèce sont présentes simultanément. Un multi-ensemble peut se formaliser par une somme formelle où l'opérateur $\oplus$ est *associatif* et *commutatif*. Par exemple

$$(a \oplus b) \oplus (c \oplus b)$$

représente un multi-ensemble (i.e., une solution chimique) contenant les éléments (i.e., les molécules) a, b et c où b apparaît en deux exemplaires. Puisque l'opération $\oplus$ est associative, on peut se passer des parenthèses et la propriété de commutativité nous permet de réordonner les élément de cette somme comme l'on veut :

$$(a \oplus b) \oplus (c \oplus b) \;=\; a \oplus b \oplus c \oplus b \;=\; a \oplus b \oplus b \oplus c \;=\; c \oplus b \oplus a \oplus b \;=\; \ldots$$

Un multi-ensemble correspond donc à un arbre où l'associativité permet d'« aplatir » les branches, et la commutativité de permuter les feuilles.

Dans une solution chimique, le mouvement brownien agite les molécules et si l'on attend suffisamment longtemps, chaque molécule a l'occasion de rencontrer et d'interagir avec n'importe quelle autre molécule de la solution. Une fois que nous avons représenté l'état d'une solution chimique comme un multi-ensemble, il est donc facile de formuler les réactions chimiques comme des règles de réécriture sur les multi-ensembles. L'associativité et la commutativité de l'opérateur $\oplus$ jouent le rôle du mouvement brownien et permettent de « regrouper » arbitrairement les éléments du multi-ensemble correspondant à une partie gauche de règle avant de l'appliquer. Par exemple, les trois règles :

$$r_1 :\; a \oplus a \to a \oplus a \oplus b \qquad r_2 :\; a \oplus b \to a \oplus b \oplus b \qquad r_3 :\; b \oplus b \to b \oplus b \oplus a$$

représentent des réactions catalytiques du second ordre entre deux molécules de type a et b (une collision de deux molécules catalyse la formation d'une troisième molécule, les deux premières molécules n'étant pas consommées dans cette réaction). Ainsi, si une réaction r_1 intervient dans l'état $a \oplus c \oplus a \oplus b$, le résultat sera l'état $a \oplus c \oplus a \oplus b \oplus b$ où un b supplémentaire a été produit. Remarquons qu'il n'est pas nécessaire que les deux occurrences de a soient côte à côte car on peut réarranger le terme pour que cela soit le cas.

Plusieurs réactions chimiques peuvent se produire en même temps, *en parallèle*, ce qui correspond à appliquer plusieurs règles simultanément sur des molécules différentes. On qualifie de *maximale parallèle* la stratégie d'application qui consiste à appliquer autant de règles que possible à un pas de temps donné. Une telle stratégie est non-déterministe : sur le multi-ensemble $a \oplus a \oplus b$ on peut appliquer r_1 ou bien r_2 mais les deux ne peuvent s'appliquer simultanément faute de ressources suffisantes. Dans ce cas, l'une des deux règles est choisie au hasard. Un pas de

réduction est ensuite itéré pour simuler l'évolution de l'état de la solution chimique. Pour prendre en compte la cinétique des réactions chimiques, plusieurs approches sont possibles afin d'ajuster la stratégie d'application des règles [30, 23].

Notons que dans cette approche, chaque molécule est explicitement représentée et chaque interaction est explicitement traitée : on parle de *simulation centrée-individu* (*agent-based* en anglais). Cette approche est à comparer avec les approches plus classiques où l'on représente la concentration de chaque espèce chimique plutôt que chaque molécule. Évidement, dans ce cas précis, l'approche centrée-individu est plus coûteuse en temps de calcul et en occupation-mémoire, mais permet de simuler finement des phénomènes complexes hors d'atteinte des approches globales, comme par exemple les fluctuations et les corrélations.

Cette formalisation abstraite de la chimie constitue un domaine de recherche appelé *chimie artificielle* [26, 25] abordant de multiples domaines d'application allant de la génération automatique des réactions de combustion [24] à l'étude des mécanismes d'auto-organisation dans l'évolution des réseaux auto-catalytiques [3].

3.1. Quelques exemples d'application

■ Un exemple simple de croissance de population

Pour illustrer la réécriture de multi-ensembles et son application à la modélisation, nous allons prendre un exemple de nature biologique : la multiplication d'un organisme monocellulaire dans un tube à essai. Nous supposerons qu'une cellule existe sous deux formes A et b : A représente une cellule mature prête à se diviser et b une cellule jeune qui va évoluer dans la forme A. Chaque division cellulaire de A donne une cellule de type A et une cellule de type b. Ces évolutions peuvent se formaliser par les deux règles :

$$\begin{aligned} r_1 : \quad A &\longrightarrow A \oplus b \\ r_2 : \quad b &\longrightarrow A \end{aligned}$$

Si l'état initial de notre tube à essai est représenté par $m_0 = A \oplus b \oplus b$, les trois premières évolutions nous donnent :

$$\begin{aligned} m_0 \to A \oplus b \oplus A \oplus A \to A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \to \\ \to A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \oplus A \oplus A \to \ldots \end{aligned}$$

La simulation de ce processus peut être utilisée pour déterminer par exemple le ratio des formes A et b dans la population après un temps donné. De plus, comme nous l'avons déjà mentionné, on peut essayer de prouver des propriétés vérifiées par tous les processus satisfaisant à ces règles d'évolution. Par exemple, Fibonacci[5] a prouvé que le ratio $\#A/\#b$ du nombre de A sur le nombre de b converge asymptotiquement vers le nombre d'or, quel que soit l'état initial.

[5] En 1602, Fibonacci s'est posé la question de savoir à quelle vitesse croissait une population de lapins dans des conditions idéales. Supposons qu'une paire de lapins, un mâle et une femelle, soit mise dans un pré. Ces lapins sont capables de se reproduire au bout d'un mois, de telle manière qu'à la fin du second mois, la femelle a engendré une autre paire de lapins. Pour simplifier, on suppose que les lapins ne meurent jamais et qu'une femelle engendre toujours, chaque mois à partir du second mois, une nouvelle paire composée d'un mâle et d'une femelle. Si on représente une paire nouvellement engendrée par b et une paire mature par A, la règle r_1 correspond à l'engendrement d'une nouvelle paire et la règle r_2 à la maturation d'une paire nouvellement créée.

■ Applications à la modélisation des réseaux d'interactions biologiques

La modélisation des réseaux d'interactions biologiques (réseau de régulation génétique, réseau de signalisation, cascade métabolique, etc.) est un domaine d'application relativement nouveau de ces techniques. Fisher et ses co-auteurs ont proposé dans [29] d'utiliser la notion de multi-ensemble pour représenter les protéines participant à une cascade d'interactions dans un chemin de signalisation. Ces travaux se sont largement développés, en particulier pour prendre en compte les différents complexes que peuvent former les protéines [27, 28].

Si la réécriture de multi-ensembles a été utilisée pour modéliser les réseaux de signalisation et les voies métaboliques, il ne faudrait pas en déduire pour autant que la cellule peut se comparer à une éprouvette contenant une soupe chimique bien mélangée. Bien au contraire, la cellule est un milieu spatialement très organisé avec des compartiments, des vésicules, des cargos, des membranes... qui permettent de localiser les différentes espèces chimiques qui interviennent (par exemple des récepteurs sont localisés sur la membrane de la cellule alors que les gènes sont localisés dans le noyau ; d'autres protéines sont ancrées et diffusent dans des membranes comme le réticulum endoplasmique). Cette localisation permet par exemple de rendre beaucoup plus efficaces certaines réactions. D'autres phénomènes, comme l'extrême densité des protéines dans le milieu intracellulaire, rendent inadéquat le modèle simple de la soupe chimique. La prise en compte de cette organisation spatiale est un des défis actuels de la modélisation des processus cellulaires [33, 34].

■ La diffusion de la chaleur dans une barre

Nous avons indiqué plus haut que les propriétés d'associativité et de commutativité permettait de déstructurer un terme afin que tout élément puisse interagir avec n'importe quel autre élément, à la manière des molécules dans une soupe chimique bien mélangée. Mais la réécriture de multi-ensembles peut être « détournée » afin de prendre en compte des informations géométriques par un codage approprié.

Le processus que nous voulons modéliser est celui de la diffusion d'un ensemble de particules le long d'une ligne. Ce problème correspond aussi à la diffusion de la chaleur dans une fine barre où chaque particule représente un quantum de chaleur. La ligne est discrétisée en une séquence de petits intervalles indexés par des entiers consécutifs. Chaque intervalle contient une ou plusieurs particules. A chaque pas de temps, une particule peut rester dans le même intervalle ou bien sauter dans l'intervalle voisin, cf. figure 1.5. On peut représenter un état de la ligne par un multi-ensemble où chaque nombre n représente une particule présente dans l'intervalle numéro n.

L'évolution du système est alors spécifiée par les 3 règles :

$$\begin{aligned} r_1 : &\quad n \longrightarrow n \\ r_2 : &\quad n \longrightarrow n-1 \\ r_3 : &\quad n \longrightarrow n+1 \end{aligned}$$

où n est un entier et où les opérations $+$ et $-$ qui apparaissent en partie droite sont les opérations arithmétiques habituelles. La règle r_2 (resp. r_3) spécifie le comportement d'une particule qui saute dans l'intervalle voisin à gauche (resp. à droite) et la règle r_1 spécifie une particule qui reste sur place.

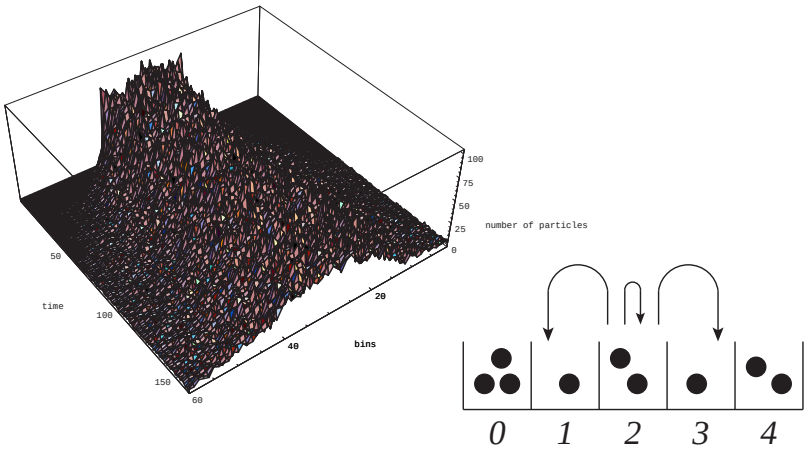

Figure 1.5. Diffusion de particules dans une barre. La figure de gauche montre l'évolution sur 160 pas de temps de 1500 particules initialement distribuées dans 20 intervalles au milieu d'une barre discrétisée en 60 intervalles. Le diagramme à droite spécifie le comportement d'une particule (cf. texte). Le nombre de particule dans un intervalle correspond à la concentration d'un produit chimique ou bien à une quantité de chaleur. Le multi-ensemble représentant l'état de la ligne (qui est discrétisée en 5 intervalles numérotés de *0* à *5*) est donné par : $0 \oplus 0 \oplus 0 \oplus 1 \oplus 2 \oplus 2 \oplus 3 \oplus 4 \oplus 4$.

3.2. Les systèmes de Păun et la compartimentalisation

Le codage précédent permet de prendre en compte une géométrie linéaire. D'autres variations ont été proposées afin de faciliter la représentation de structures biologiques plus complexes, comme par exemple l'imbrication des membranes et des compartiments dans une cellule : l'élément d'un multi-ensemble peut être une molécule ou bien un autre multi-ensemble qui à son tour peut contenir des molécules et des multi-ensembles.

De telles imbrications sont étudiées dans le formalisme des *systèmes de Păun* (P-systèmes) [35] où la réécriture classique de multi-ensembles est étendue par la notion de *membrane*. Une membrane est une imbrication de compartiments représentée, par exemple, par un diagramme de Venn[6] sans intersection et avec un unique sur-ensemble : la *peau* du système, cf. figure 1.6.

Des objets sont placés dans chaque région délimitée par une membrane et ils évoluent suivant divers mécanismes : un objet (ou un multi-ensemble d'objets) peut se transformer en d'autres objets, mais peut aussi passer à travers une membrane, provoquer la dissolution d'une membrane ou bien sa création. La figure 1.7 donne quelque exemples. Formellement, un tel système peut se spécifier en utilisant plusieurs opérations $\oplus, \oplus', \oplus'', \ldots$ correspondant chacune à une certaine membrane. Ces opérations sont associatives et commutatives, mais ne sont pas associatives entre elles (afin de garder les membranes séparées).

[6] Les diagrammes de Venn, dû au logicien anglais de même nom, permettent de visualiser les opérations ensemblistes en représentant les ensembles par la surface délimitée par une courbe fermée.

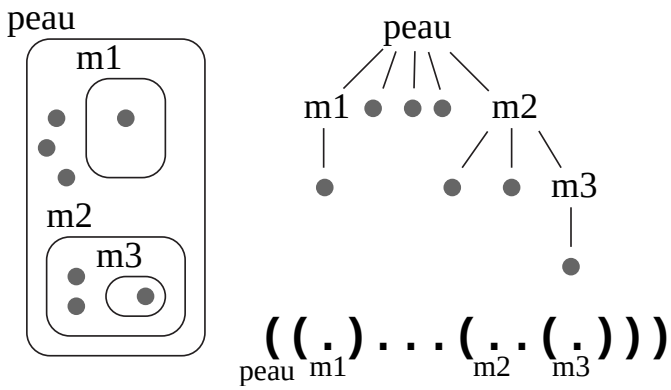

Figure 1.6. Systèmes de Păun. Une imbrication de membrane peut se représenter par un diagramme de Venn sans intersection et avec un unique sur-ensemble, par un arbre d'imbrication ou encore par un mot bien parenthésé.

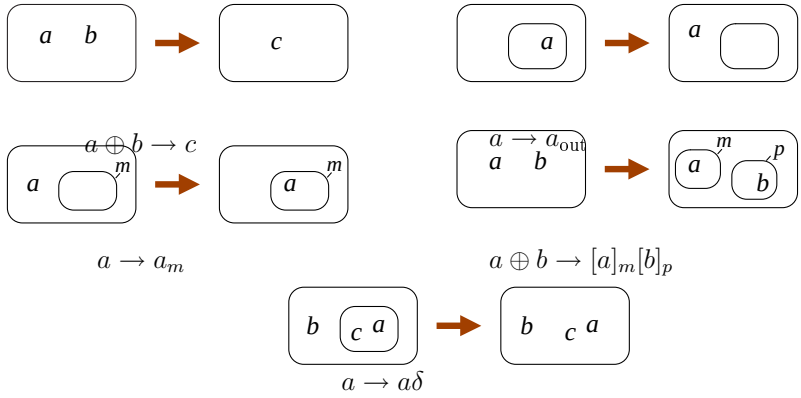

Figure 1.7. Exemple de règle d'évolution dans un Système de Păun. Le symbole δ en partie droite d'une règle implique la dissolution de la membrane englobante. Les membranes de destination sont indiquées en indice (les membranes sont nommées). La membrane enclosante peut toujours se référer par le nom "out".

3.3. Une parenthèse : application à la programmation parallèle

Le dialogue entre l'informatique et les autres disciplines scientifiques ne se fait pas à sens unique. Voici un exemple. Inspirés par la métaphore chimique, les informaticiens ont utilisé la réécriture de multi-ensembles non seulement pour simuler des réactions chimiques ou des processus biologiques, mais aussi comme langage de programmation parallèle. C'est le langage GAMMA [36] qui le premier a développé cette idée. Voici un exemple de programme parallèle particulièrement élégant :

$$x \oplus y \;/\; (x \bmod y = 0) \;\longrightarrow\; y$$

Cette règle spécifie qu'il faut remplacer le couple de nombres x, y par y dès que la condition « y divise x » est vérifiée (la condition s'écrit derrière le symbole /). Si on applique cette règle autant que faire se peut à un multi-ensemble initialement composé de tous les nombres entre 2 et n, on obtient un multi-ensemble dans lequel cette règle ne peut s'appliquer (car la condition ne peut y être vérifiée) et qui contient donc tous les nombres premiers jusqu'à n.

Dans le programme précédent, il n'y a aucune trace d'un séquencement artificiel des calculs : l'application de la règle se fait dans n'importe quel ordre. Notons que

le parallélisme provient de l'application simultanée des règles et que le « déroulement » du programme consiste simplement à itérer l'application des règles jusqu'à obtenir une forme normale (un point fixe). Ces programmes sont non-déterministes, sauf si les règles de réécriture sont confluentes : dans ce cas, quand le programme termine, on a bien un résultat parfaitement déterminé même si les valeurs intermédiaires calculées au cours de l'exécution du programme peuvent être différentes (on parle de déterminisme du résultat malgré le non-déterminisme de l'exécution).

4. Les systèmes de Lindenmayer et la croissance des structures linéaires

Dans la section précédente nous avons considéré un processus de réécriture sur des termes associatifs et commutatifs, ce qui permettait de modéliser une « soupe chimique ». Dans cette section, nous nous intéressons à des termes associatifs : ces termes correspondent alors à des séquences et on parle de *réécriture de chaînes* (de symboles). Les travaux de Chomsky sur les grammaires formelles [37] ont marqué le début de très nombreux travaux sur la réécriture de chaînes. Ce sont ces travaux qui sont à la source des développements sur la syntaxe, la sémantique et les langages formels en informatique. Les grammaires sont des formalismes génératifs (i.e. qui permettent de construire des familles d'objets) en engendrant des ensembles de *phrases* : une phrase est une séquence de *symboles* engendrée par des réécritures successives. L'ensemble des phrases que l'on peut générer est un *langage*.

Le biologiste Aristid Lindenmayer (1925–1989) a introduit en 1968 un nouveau type de réécriture de chaînes pour servir de fondation à une théorie formelle du développement biologique [38] : les systèmes de Lindenmayer, ou en abrégé, les L-systèmes. La différence principale avec les grammaires de Chomsky tient en la stratégie d'application des règles. Dans les grammaires de Chomsky, il n'y a qu'une seule réécriture qui s'applique à la fois alors que dans les L-systèmes les réécritures ont lieu en parallèle, substituant à chaque étape tous les symboles d'une phrase. Lindenmayer justifiait cette stratégie par l'analogie avec le développement cellulaire : toutes les cellules d'un organisme se divisent indépendamment et en parallèle.

La motivation des L-systèmes est de construire un objet complexe (comme une plante) en remplaçant successivement des parties d'un objet plus simple, au moyen de règles de réécriture. Les symboles sont interprétés comme des composants d'un organisme vivant, comme par exemple des cellules ou des organes, plutôt que comme des mots. Les L-systèmes ont trouvé de nombreuses applications dans la modélisation de la croissance des plantes, mais aussi en infographie avec la génération de courbes fractales ou de plantes virtuelles.

4.1. Croissance d'une structure filamentaire

Un exemple simple de L-système est celui qui décrit la croissance d'une cyanobactérie *Anabaena Catenula*. Cette algue croît en filament composé, pour notre exemple, de quatre types de cellules `G`, `g`, `D` et `d` qui s'interprètent de la manière suivantes : `G` et `D` sont des cellules matures aptes à se diviser ; `g` et `d` sont des cellules quiescentes. Par ailleurs les cellules sont polarisées : `D` et `d` sont polarisées vers la droite dans le filament ; `G` et `g` sont polarisées vers la gauche.

Quand on examine un filament, les différentes cellules ne se succèdent pas dans n'importe quel ordre. En fait, le système de réécriture ci-dessous permet de générer des séquences très proches de ce que l'on peut observer dans la nature.

$$\begin{aligned} \texttt{D} &\longrightarrow \texttt{Gd} \\ \texttt{G} &\longrightarrow \texttt{gD} \\ \texttt{d} &\longrightarrow \texttt{D} \\ \texttt{g} &\longrightarrow \texttt{G} \end{aligned}$$

$$\begin{aligned} t_1 &: \texttt{D} \\ t_2 &: \texttt{Gd} \\ t_3 &: \texttt{gDD} \\ t_4 &: \texttt{GGdGd} \\ t_5 &: \texttt{gDdGDgDD} \end{aligned}$$

Les dérivations à droite montrent qu'à chaque étape tous les symboles sont réécrits en parallèle à l'aide des règles de gauche.

Ce mécanisme de base admet de nombreuses variations qui ont pour objectif d'étendre l'expressivité du formalisme. Une des extensions les plus importantes est de décorer les symboles par des attributs, comme par exemple un nombre représentant une taille ou bien la concentration d'un produit chimique.

La figure 1.8 illustre l'utilisation d'une telle extension dans un modèle plus réaliste de la croissance d'*Anabaena*. Plutôt que de considérer des cellules matures et des cellules quiescentes, chaque cellule possède une taille qui croît avec le temps. Par ailleurs, dans un milieu privé d'azote, certaines cellules se spécialisent : ce sont les hétérocystes. Wilcox *et al.* [44] ont supposé qu'une cellule se différenciait en hétérocyste sous l'action de deux substances chimiques, un activateur et un inhibiteur, qui diffusent dans le filament et qui réagissent entre elles. Ce modèle de réaction-diffusion permet d'expliquer l'apparition d'une cellule hétérocyste isolée toutes les n cellules végétatives, ce que l'on observe dans la nature avec un entier n relativement constant. Prunsinkiewicz et Hammel [43] ont spécifié et simulé ce système à l'aide d'un L-système, réalisant ainsi la simulation d'une réaction-diffusion dans un médium en croissance (le filament), un important exemple de $(\text{SD})^2$.

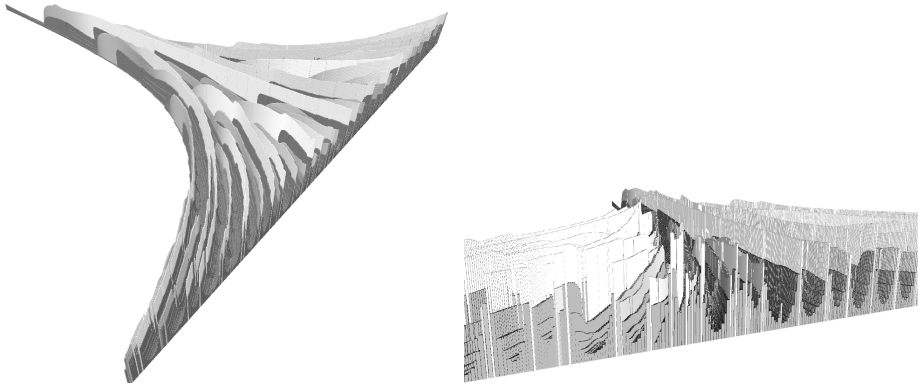

Figure 1.8. Différenciation des hétérocystes dans un filament d'*Anabaena*. La figure de gauche représente le même diagramme que la figure de droite, mais vu sous un angle différent. Cette représentation graphique, appelée extrusion temporelle, a été introduite dans [43]. Dans ce diagramme le temps va du coin supérieur gauche au coin inférieur droit. Chaque « tranche » représente les cellules d'un filament à un instant donné. La hauteur de chaque cellule représente la concentration en activateur tout comme la couleur de la cellule (du noir au blanc). Les cellules en noir représentent les cellules végétatives. La différenciation intervient quand l'activateur dépasse un certain seuil.

4.2. Croissance d'une structure arborescente

Il est facile de représenter une structure arborescente par une chaîne en introduisant deux symboles servant de « parenthèses ». La réécriture de telles chaînes présente de subtiles différences avec la réécriture directe des termes. C'est la voie utilisée par les L-systèmes pour représenter la structure arborescente d'une plante et ses règles de développement. L'exemple ci-dessous est caricatural et ne correspond à la croissance d'aucune plante naturelle. Il permet cependant d'illustrer la puissance de l'approche.

Supposons qu'une plante soit constituée de deux types de « branche » : les branches simples `b` et les branches portant un bourgeon `B`. D'une année à l'autre, une branche avec un bourgeon perd son bourgeon pour devenir une branche simple. On a donc la règle $\mathtt{B} \rightarrow \mathtt{b}$. Une branche simple croît et produit une partie de plante composée d'un axe constitué de 3 branches simples et de 2 branches avec des bourgeons partant sur les côtés à 1/3 et à 2/3 de la hauteur. Cette spécification se traduit par la règle :

$$\mathtt{b} \longrightarrow \mathtt{b}\langle q\mathtt{B}\rangle\mathtt{b}\langle p\mathtt{B}\rangle\mathtt{b}$$

Dans cette règle, on utilise les parenthèses $\langle$ et $\rangle$ pour représenter le développement des axes latéraux. On utilise les symboles additionnels p et q pour indiquer le développement à droite ou à gauche de ces axes.

Au début des années 1980, P. Prusinkiewicz introduit une interprétation graphique des mots produits par un L-système [39]. Cette interprétation est fondée sur la notion de tortue graphique utilisée par exemple dans le langage LOGO. Cette interprétation permet de visualiser directement la structure des objets décrits par un mot engendré par le L-système. Ainsi, on utilise les dérivations successives d'un L-système pour représenter les états successifs d'une plante qui se développe ou bien pour dessiner les courbes successives qui tendent vers une courbe fractale.

Un état de la tortue graphique est un triplet (x, y, θ) où (x, y) représente la position courante de la tortue dans un repère cartésien et où θ représente l'orientation de la tortue. Cette orientation est interprétée comme l'angle que fait le corps de la tortue avec l'axe horizontal. La tortue exécute des commandes de déplacement représentées par les symboles d'un mot.

- Dans notre exemple, le symbole `b` correspond à avancer d'une longueur Δ. Si l'état courant était (x, y, θ), après lecture du symbole `b`, il devient $(x + \Delta\cos\theta, y + \Delta sin\theta, \theta)$.
- Le symbole `B` est interprété de la même manière à ceci près qu'après le tracé du segment correspondant, on trace aussi un cercle centré sur la position courante.
- Les symboles $\langle$ et $\rangle$ permettent respectivement de sauvegarder la position courante et de la restaurer. Cette sauvegarde se fait dans une pile. Quand elle rencontre le symbole $\rangle$, la tortue « saute » à la position correspondant à la parenthèse ouvrante.
- Enfin le symbole p (resp. q) incrémente (resp. décrémente) l'angle courant θ d'un angle préfixé.

Avec cette interprétation graphique, les 3 premières dérivations d'une branche simple sont illustrées figure 1.9.

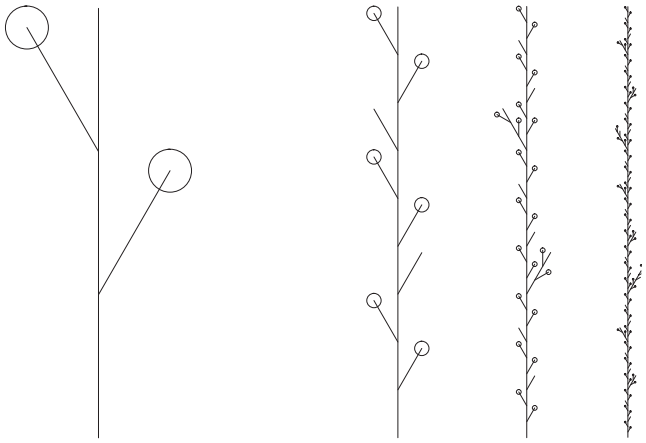

Figure 1.9. Représentation graphique des 3 premières dérivations d'un système de Lindenmayer. L'état initial est donné par le mot $\mathtt{b}\langle q\mathtt{B}\rangle\mathtt{b}\langle p\mathtt{B}\rangle\mathtt{b}$ et les règles de dérivation sont les deux règles données dans le texte. L'échelle de la représentation de chaque « plante » est différente.

5. Au-delà des structures linéaires : calculer une forme pour la comprendre

Les L-systèmes se sont révélés parfaitement adaptés à la modélisation de la croissance des plantes [39] : ils permettent de définir de manière particulièrement compacte et synthétique la création de la forme complexe d'une plante et surtout, dans leurs extensions récentes, de coupler le processus de création de la forme aux processus physico-chimiques qui prennent place dans cette forme.

Cependant, les L-systèmes sont adaptés à la représentation de formes linéaires (filaments ou arborescences) et leur utilisation pour la construction de formes plus complexes (graphes quelconques, surfaces, volumes) repose sur des codages arbitraires qui deviennent rapidement inextricables. C'est pourquoi les chercheurs tentent de concevoir d'autres formalismes plus adaptés. Cette recherche de formalismes adaptés à la spécification des processus de développement va bien au-delà de la seule simulation, pour deux raisons : ces formalismes peuvent remplir un vide conceptuel en biologie et ont potentiellement un impact épistémologique énorme. Par ailleurs, leur application pourrait s'étendre bien au-delà des seuls objets biologiques.

■ Simulation et explication

S'appuyant d'abord sur des modèles purement physiques (les croissances osmotiques avec S. Leduc, les formes optimales avec d'Arcy Thompson, les processus de réaction-diffusion avec Turing, etc.), puis purement génétique (avec des notions comme l'action génique ou le programme génétique) les différents formalismes proposés au cours du siècle précédent pour spécifier les processus de développement ont rempli un vide conceptuel et ils ont modifié la perception de ce qui a valeur d'explication pour un biologiste [5].

À titre d'exemple, les progrès informatiques et les données produites par la biologie permettent la simulation de certains processus de développement avec des prédictions qui peuvent être validées par un retour à l'expérience [10]. Ainsi très

récemment, plusieurs modèles [6, 11] concernant le niveau cellulaire de la croissance du méristème (la partie en croissance de la plante) ont permis de reproduire les motifs phyllotaxiques caractéristiques qui sont observés dans la nature et de les relier à la circulation de l'auxine (une hormone végétale) dans cet organe. L'accumulation d'auxine déclenche le développement de nouveaux organes, entités qui modifient la forme du méristème et donc les flux d'auxine : on a là un merveilleux exemple de (SD)^2.

Cependant, ces simulations, aussi prédictives soient-elles, n'auront une valeur explicative que si elles permettent d'exprimer les processus du développement sous une forme accessible à l'esprit humain, afin de les analyser et de raisonner dessus. En effet, de quel type de compréhension pouvons nous nous targuer si nous nous contentons d'être les observateurs d'une suite de calculs complexes ? Autant nous contenter de les observer dans la nature, plutôt que de les reproduire dans un calculateur.

La morphogénèse informatique permet de fixer un cadre formel où il devient possible de parler rigoureusement de programme génétique, de mémoire, d'information, de signal, d'interaction, d'environnement... et d'articuler ces notions avec une vision complètement mécaniste des processus du développement. Elle introduit la notion de *calcul* comme schéma explicatif dans la modélisation du développement. Mais si l'embryon peut se déduire par un calcul à partir de la description de l'œuf et des interactions avec l'environnement, il faut concevoir l'embryon à la fois comme le résultat d'un calcul, et à la fois comme une partie du calculateur qui calcule ce résultat. Cette problématique est étudiée en informatique (programme réflexif, programme méta-circulaire). L'avenir nous dira si ces notions sont à même de permettre la saisie du vivant dans ce qu'il a de plus spécifique, le développement.

■ Donner une forme à une population d'agents autonomes

La modélisation des processus de développement est importante pour le biologiste, mais elle l'est aussi pour l'informaticien toujours à la recherche de nouveaux modèles de calcul et la biologie est à l'évidence une grande source d'inspiration.

Les modèles de calculs sont contraints par les particularités d'un modèle matériel ou bien inspirés par une métaphore de ce que doit être un calcul. Aujourd'hui, de nouveaux supports matériels du calcul sont étudiés. On connaît l'expérience d'Adleman en 1994 [22], qui démontre qu'on peut résoudre un problème d'optimisation combinatoire (la recherche d'un chemin Hamiltonien dans un graph, un problème NP-complet similaire à celui du voyageur de commerce) dans une éprouvette avec des molécules d'ADN. Mais d'autres possibilités font l'objet de recherches très actives : on espère ainsi utiliser la croissance de colonies de bactéries, la diffusion de réactifs chimiques, l'auto-assemblage de biomolécules... pour calculer. La programmation de ces nouveaux supports d'exécution pose des problèmes certains et motive le développement de langages et d'une algorithmique adaptés qui permettraient d'utiliser une population immense d'entités autonomes (des biomolécules, des virus, des cellules) qui interagissent localement de manière irrégulière et non-fiable, pour construire et faire émerger un calcul fiable (une forme).

Mais, sans devoir recourir à des machines biologiques construites avec les biotechnologies, les mécanismes offerts par un langage de programmation, ou de nouveaux algorithmes, peuvent être directement inspirés par une métaphore biologique. Par exemple, les algorithmes évolutionnistes s'inspirent des mécanismes étudiés par la

théorie de l'évolution même s'ils s'exécutent sur des machines électroniques comme les ordinateurs actuels.

Dans cet ordre d'idée, les formalismes permettant de saisir conceptuellement les mécanismes du développement pourraient bien servir à renouveler la notion de programme en suggérant de nouvelles approches dans le développement de très grands logiciels, pour la spécification de leur architecture, l'interconnexion de leurs parties, en offrant de nouveaux mécanismes pour cacher l'information inutile, abstraire les détails ou capitaliser et réutiliser le code. Car les informaticiens recherchent activement pour leurs logiciels des propriétés habituellement attribuées au vivant : l'autonomie, l'adaptabilité, l'auto-réparation, la robustesse, l'auto-organisation. On voit donc que le dialogue que l'informatique entretient avec la biologie, dialogue ambigu, dialogue fertile, n'est pas près de cesser.

Bibliographie

■ **Modélisation et biologie :**

[1] H. Atlan. *La Fin du « tout-génétique » ? Vers de nouveaux paradigmes en biologie.* INRA, 1999.

[2] M. Eigen et P. Schuster. *The Hypercycle : A Principle of Natural Self-Organization.* Springer, 1979.

[3] W. Fontana et L. Buss. "The arrival of the fittest" : Toward a theory of biological organization. *Bulletin of Mathematical Biology*, 1994.

[4] E. Fox Keller. *Le rôle des métaphores dans les progrès de la biologie.* Les Empecheurs Penser en Rond, 1999.

[5] E. Fox Keller. *Expliquer la vie (modèles, métaphores et machines en biologie du développement).* Gallimard, 2004.

[6] P. Barbier de Reuille, I. Bohn-Courseau, K. Ljung, H. Morin, N. Carraro, C. Godin et Jan Traas. Computer simulations reveal properties of the cell-cell signaling network at the shoot apex in Arabidopsis. *PNAS*, 103(5) :1627–1632, 2006.

[7] G. Israel. *La mathématisation du réel.* Seuil, 1996.

[8] Y. Itkis. *Control Systems of Variable Structure.* Wiley, 1976.

[9] R. Paton, éditeur. *Computing With Biological Metaphors.* Chapman & Hall, 1994.

[10] E. Coen, A.-G. Rolland-Lagan, M. Matthews, J. Andrew Bangham et P. Prusinkiewicz. The genetics of geometry. *PNAS*, 101(14) :4728–4735, 2004.

[11] R. S. Smith, S. Guyomarc'h, T. Mandel, D. Reinhardt, C. Kuh lemeier et P. Prusinkiewicz. A plausible model of phyllotaxis. *PNAS*, 103(5) :1301–1306, 2006.

[12] F. J. Varela. *Principle of Biological Autonomy.* McGraw-Hill/Appleton & Lange, 1979.

[13] J. Von Neumann. *Theory of Self-Reproducing Automata.* Univ. of Illinois Press, 1966.

[14] INTERSTICE (site de présentation des activités de recherche dans le domaine des STIC). *Dossier sur la bio-informatique.* `http://interstices.info/display.jsp?id=c_6474`

■ **Systèmes de réécriture :**

[15] N. Dershowitz. *A Taste of Rewrite Systems*, volume 693 des *Lecture Notes in Computer Science*, pages 199–228. Springer Verlag, 1993.

[16] N. Dershowitz et J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapitre Rewrite systems, pages 244–320. Elsevier Science, 1990.

[17] The Maude project. Maude home page, 2002. `http://maude.csl.sri.com/`.

[18] The Elan project. Elan home page, 2002. `http://www.loria.fr/equipes/protheo/SOFTWARES/ELAN/`.

■ **SR, extension et simulation des (SD)^2 :**

[19] J.-L. Giavitto et O. Michel. The topological structures of membrane computing. *Fundamenta Informaticae*, 49 :107–129, 2002.

[20] J.-L. Giavitto et O. Michel. Modeling the topological organization of cellular processes. *BioSystems*, 70(2) :149–163, 2003.

[21] J.-L. Giavitto. Invited talk : Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In *Rewriting Technics and Applications (RTA'03)*, volume 2706 des *Lecture Notes in Computer Science*, pages 208–233, Valencia, Juin 2003. Springer.

■ **Réécriture de multi-ensembles :**

[22] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187) :1021–4, Nov. 1994.

[23] O. Bournez et M. Hoyrup. Rewriting logic and probabilities. In R. Nieuwenhuis, éditeur, *14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*, volume 2706 des *Lecture Notes in Computer Science*, pages 61–75. Springer, June 9-11 2003.

[24] O. Bournez, H. K. Guy-Marie Côme, Valérie Conraud et L. Ibanescu. A rule-based approach for automated generation of kinetic chemical mechanisms. In R. Nieuwenhuis, éditeur, *14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*, volume 2706 des *Lecture Notes in Computer Science*, pages 30–45. Springer, June 9-11 2003.

[25] P. Dittrich. Artificial chemistry webpage. `ls11-www.cs.uni-dortmund.de/achem`, 2001.

[26] P. Dittrich, J. Ziegler et W. Banzhaf. Artificial chemistries - a review. *Artificial Life*, 7(3) :225–275, 2001.

[27] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer et J. Sonmez. Pathway logic : Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[28] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, et C. Talcott. Pathway logic : Executable models of biological networks. In *Fourth International Workshop on Rewriting Logic and Its Applications (WRLA'2002)*, volume 71 des *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[29] M. Fisher, G. Malcolm et R. Paton. Spatio-logical processes in intracellular signalling. *BioSystems*, 55 :83–92, 2000.

[30] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81, pages 2340–2361, 1977.

[31] T. Head. Formal language theory and DNA : an analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology*, 49 :737–759, 1987.

[32] T. Head. *Lindenmayer Systems : Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, chapitre Splicing schemes and DNA, pages 371–383. Springer, 1992. Republié dans : *Nanobiology*, 1(1992)335-342.

[33] C. Lemerle, B. Di Ventura et L. Serrano. Space as the final frontier in stochastic simulations of biological systems. *Minireview. FEBS Letters*, 579 :1789–1794, février 2005.

[34] K. Takahashi, S. Nanda Vel Arjunan et M. Tomita. Space in systems biology of signaliting pathways - towards intracellular molecular crowding in silico. *Minireview. FEBS Letters*, 579 :1783–1788, février 2005.

[35] G. Păun. *Membrane Computing. An Introduction.* Springer-Verlag, 2002.

[36] J.-P. Banatre, A. Coutant et D. L. Metayer. A parallel machine for multiset transformation and its programming style. *Future Generation Computer Systems*, 4 :133–144, 1988.

■ **Systèmes de Lindenmayer :**

[37] N. Chomsky, éditeur. *Syntactic structures.* Mouton & Co., La Hague, 1957.

[38] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18 :280–315, 1968.

[39] P. Prusinkiewicz, A. Lindenmayer, J. Hanan *et al. The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[40] P. Prusinkiewicz et J. Hanan. Visualization of botanical structures and processes using parametric L-systems. In D. Thalmann, éditeur, *Scientific visualization and graphics simulation*, pages 183–201. J. Wiley & Sons, Chichester, 1990.

[41] P. Prusinkiewicz. Modeling of spatial structure and development of plants : a review. *Scientia Horticulturae*, 74 :113–149, 1998.

[42] P. Prusinkiewicz. A look at the visual modeling of plants using L-systems. *Agronomie*, 19 :211–224, 1999.

[43] M. Hammel et P. Prusinkiewicz. Visualization of developmental processes by extrusion in space-time. In *Proceedings of Graphics Interface '96*, pages 246–258, 1996.

[44] M. Wilcox, G. J. Mitchison et R. J. Smith. Pattern formation in the blue-green alga, *Anabaena*. I. Basic mechanisms. *Journal of Cell Science*, 12 :707–723, 1973.