Travaux effectués

Antoine Spicher

1 Résumé

Mon travail de thèse s'est déroulé dans le cadre du projet MGS de l'équipe *LIS* (**L**angages, **I**nteractions et **S**imulations) du laboratoire IBISC (**I**nformatique, **B**iologie **I**ntégrative et **S**ystèmes **C**omplexes), FRE 2873 du CNRS, Université d'Évry. Ce laboratoire est une nouvelle unité de recherche issue de la fusion au 1er janvier 2006 du **La**boratoire des **M**éthodes **I**nformatiques (LaMI - UMR 8042 du CNRS) dans lequel j'ai débuté ma thèse, et du **L**aboratoire des **S**ystèmes **C**omplexes (LSC - FRE 2494 du CNRS).

Mes travaux de thèse portent sur le développement d'un langage de programmation dédié à la modélisation et la simulation de systèmes dynamiques complexes, en particulier dans le domaine de la biologie. Ils se fondent sur des notions empruntées à la topologie algébrique. Les apports de mes travaux concernent :

- *la programmation* : nouvelles structures de données, définition par cas de fonctions sur des structures de données non-algébriques, programmation par règles et stratégies stochastiques, formalisation topologique des structures de données, sémantique (dans un style naturel) ;
- et *la simulation informatique de systèmes complexes* : biologie du développement, interactions moléculaires, mobilité cellulaire, processus de diffusion limitée par agrégation sur des géométries arbitraires, etc.

Dans la suite de cette section, je détaille le contexte du projet MGS puis mes travaux de thèse.

2 Le projet MGS et les systèmes dynamiques à structure dynamique

Le projet MGS, cadre de mon travail de thèse, poursuit deux objectifs complémentaires :

1. étudier et développer l'apport de *notions* et d'*outils* de nature *topologique* dans les langages de programmation ;
2. appliquer ces notions et ces outils à la conception et au développement de nouvelles structures de données et de contrôle à la fois expressives et efficaces pour la modélisation et la simulation de *systèmes dynamiques à structure dynamique* (abrégé en $(SD)^2$ dans la suite).

Ces études se concrétisent par le développement d'un langage de programmation expérimental et par son application à la modélisation et à la simulation de systèmes dynamiques, en particulier dans le domaine de la biologie et de la morphogenèse. Ce langage est également nommé MGS.

2.1 La notion de système dynamique à structure dynamique

La simulation des systèmes dynamiques. Intuitivement, un système dynamique est une façon formelle de décrire comment un point (l'état du système) se déplace dans un *espace des phases* (l'espace de tous les états possibles du système). Il faut donc spécifier une règle, la *fonction d'évolution*, décrivant la trajectoire de ce point. Il existe de nombreux formalismes qui sont utilisés pour décrire un système dynamique : équations différentielles ordinaires (EDO), équations différentielles partielles (EDP), couplage discret d'équations différentielles, itération de fonctions, automates cellulaires... suivant la nature discrète ou continue du temps et de l'espace ainsi que des valeurs utilisées lors de la modélisation.

Les systèmes dynamiques à structure dynamique. En suivant l'analyse qui a été développée dans [GGMP02], on peut remarquer que de nombreux systèmes biologiques peuvent être vus comme des systèmes dynamiques dans lesquels non seulement la valeur des variables d'état, mais aussi l'*ensemble* de ces variables d'état *et/ou* la fonction d'évolution, changent au cours du temps. Autrement dit, l'espace des phases doit être conçu comme *une observable* du système. Ce sont des $(SD)^2$ suivant la terminologie introduite dans [GM01b, GM01a].

La notion de $(SD)^2$ est particulièrement évidente en biologie du développement. Mais cette notion est centrale dans plusieurs formalismes proposés pour la modélisation de systèmes biologiques : les *hyper-cycles* [ES79], les *systèmes autopoïetiques* [Var79], les *variable structure systems* [Itk76, HGH93], les *developmental grammars* [MSR91] ou encore les *organisations* de [FB94].

Cependant, ce type de systèmes ne se rencontre pas uniquement en biologie : la modélisation de réseaux dynamiques (internet, réseaux mobiles), les phénomènes de morphogenèse en physique (croissance dans un médium dynamique), la mécanique des milieux élastiques ou des systèmes déformables, la croissance des villes ou encore les réseaux sociaux regorgent d'exemples de $(SD)^2$.

Quels langages pour simuler les $(SD)^2$? La spécification informatique des $(SD)^2$ pose un problème : *quel est le langage adapté à la définition d'une fonction d'évolution qui porte sur un état dont on ne peut décrire la structure à l'avance ?* Le système ne peut pas être décrit simplement de manière globale mais uniquement par un ensemble d'**interactions** [Rau03] locales entre des entités plus élémentaires qui composent le système. Notre problème est de définir ces entités élémentaires et leurs interactions.

L'approche adoptée dans le projet MGS est de développer un cadre permettant de représenter l'état d'un système dynamique à partir de sa structure spatiale, c'est la notion de *collection topologique*, et de développer les outils permettant de définir localement des fonctions d'évolution sur ces nouvelles données, c'est la notion de *transformation*.

Une collection topologique est un ensemble de valeurs muni d'une relation de voisinage ; une transformation est une fonction définie par cas sous la forme de règles de réécriture s'appuyant sur la notion de voisinage.

2.2 Un langage de programmation non conventionnel

Les notions de collection topologique et de transformation sont initialement motivées par les problèmes particuliers posés par la simulation des systèmes dynamiques à structure dynamique en biologie. Mais ces notions peuvent s'intégrer dans un langage fonctionnel classique (comme ML) où elles ouvrent d'intéressantes perspectives qui justifient à elles seules leur étude :

- ces notions offrent un *cadre uniforme* pour spécifier et manipuler des structures de données qui incluent et étendent la notion de type algébrique ;
- elles permettent d'étendre la *définition par cas*, un mécanisme puissant de spécification de fonction, à tous les types de données (par exemple aux tableaux) ;
- elles offrent un cadre alternatif à la notion de *polytypisme* [Jan00] qui n'est pas restreint aux types de données algébriques ;
- elles apportent une *nouvelle notion de réécriture* qui ne se réduit pas aux approches existantes et qui unifient plusieurs modèles de calcul.

La vision topologique adoptée par MGS permet d'unifier plusieurs modèles de calcul : le calcul chimique [BL86, BFL01], les systèmes de Păun [Pău00, Pău01], les systèmes de Lindenmayer [RS92, PLH+90] et les automates cellulaires [vN66]. Le point de vue qui nous intéresse ici est celui des langages de programmation : à chacun de ces modèles de calcul correspond une classe de langages de programmation ; nous ne nous intéressons pas à l'étude de la complexité des algorithmes exprimés dans ces modèles de calcul mais à l'expressivité permise par les constructions propres à ces modèles dans un langage associé.

Ces quatre paradigmes se présentent tous comme des langages de règles et on peut les décrire comme des formes particulières de *réécriture* (par réécriture nous entendons ici le mécanisme qui consiste à substituer une partie par une autre dans un objet). L'approche topologique permet de définir une notion générale d'organisation et de parties, et de décrire ces modèles de calcul comme des cas particuliers d'un mécanisme général de substitution de parties dans une organisation spatiale.

3 Présentation de mes travaux de thèse

Mon travail de recherche a consisté à étendre les collections topologiques standards de MGS vers des constructions topologiques de dimension arbitraire. La notion de réécriture locale de collection topologique, concrétisée à travers le concept de transformation, a été également étendue afin d'autoriser les modifications topologiques de ces structures arbitrairement complexes.

Plus précisément, mes travaux se sont organisés suivant trois directions :

1. Le premier axe de recherche concerne le développement de la notion de collection topologique. Ce développement poursuit deux objectifs : généraliser cette notion afin de prendre en compte

les besoins en modélisation, en particulier la représentation d'objets spatiaux complexes de dimension arbitraire, et fonder théoriquement cette notion à partir de celle de complexe cellulaire abstrait développée en topologie algébrique.

2. La seconde direction de recherche concerne la notion de transformation et la sémantique formelle des programmes MGS. Ce travail a permis de développer la manipulation déclarative de collections topologiques de dimension arbitraire, d'étendre le langage de motifs de chemins des règles d'une transformation, de concevoir un nouveau langage de motifs plus puissant afin de filtrer des sous-collections de forme arbitraire et de développer des stratégies stochastiques d'application des règles d'une transformation.
3. Enfin, nous avons validé par de nombreux exemples significatifs les nouvelles constructions que nous avons conçues et implantées dans l'interprète MGS. Ces applications sont pour la plupart des exemples non triviaux de systèmes dynamiques à structure dynamique. La variété des domaines abordés, le volume et l'importance des exemples traités permettent de juger de la pertinence des constructions proposées et mettent en évidence le gain en expressivité apporté par l'approche topologique.

3.1 La formalisation des collections topologiques de dimension arbitraire

Je me suis inspiré de concepts de topologie algébrique pour élaborer de façon formelle la notion de collection topologique. Il s'agit d'une structure de données adaptée à la représentation de champs sur des organisations spatiales discrètes et arbitrairement complexes, suffisamment souple pour autoriser les modifications locales de ces organisations et offrant la notion de dimension. J'ai utilisé les concepts de complexe cellulaire abstrait et de complexe de chaînes pour respectivement représenter des espaces complexes comme l'agencement de briques élémentaires, et associer des valeurs à la structure obtenue [Mun84].

J'ai ensuite décrit, dans une version simplifiée, le langage MGS développé dans le projet. Il s'agit d'un langage fonctionnel étendu avec les collections topologiques et les *transformations*, une forme de réécriture locale des collections topologiques. J'ai défini la sémantique de ce langage dans le style de la *sémantique naturelle* [Kah87] afin de rendre compte des trois mécanismes fondamentaux mis en jeu dans une transformation :

- *le filtrage d'une sous-collection :* avec deux langages de motifs (motifs de chemin et motifs de patch) ;
- *les stratégies d'application des règles :* synchrone/asynchrone avec ou sans priorité d'une part, et probabiliste d'autre part (en particulier une stratégie stochastique standard et une stratégie pour les systèmes à évènements discrets) ;
- *la reconstruction* : fondée sur une algèbre d'opérations sur les collections topologiques.

Cette sémantique est la première qui rende compte de ces trois étapes simultanément. Une version stochastique de la sémantique associe à chaque expression une densité de probabilité sur les valeurs. De plus, la sémantique que nous avons développée a été utilisée pour démontrer formellement que dans le cas d'une stratégie maximale parallèle, si la transformation ne s'applique pas, c'est qu'il n'existe aucune sous-collection pouvant être filtrée par un motif d'une règle de la transformation.

3.2 Des applications

Une part importante de mon travail a consisté à éprouver et valider les concepts développés par des applications spécifiques. Pour cela, il a fallu tout d'abord implanter les structures de données et les transformations proposées dans un interprète. Nous avons ensuite développé et simulé un grand nombre de modèles exprimés dans le langage MGS. Outre l'illustration des concepts proposés, nous avons profité de l'expérience gagnée pour déterminer les limites de notre formalisme et pour apporter les nouveautés et les améliorations nécessaires.

Ces applications répondent à différents types de problématiques de simulation des $(SD)^2$:

- *Modifications topologiques* : comment calculer déclarativement un espace complexe de dimension arbitraire à partir de règles de construction locales ? Les *patches*, un type particulier de transformation que j'ai développé, permettent de telles opérations.
- *Calcul numérique* : comment exprimer simplement la résolution numérique d'équations différentielles sur des organisations spatiales arbitraires ? Les *transformations de chemins* fournissent les moyens de programmer localement et de façon implicite (dans le sens où les coordonnées des objets spatiaux n'apparaissent pas) de tels algorithmes.
- *Modèles et simulations stochastiques* : comment spécifier de manière déclarative l'évolution d'un phénomène connaissant les lois de probabilité qui le caractérisent ? Les stratégies d'ap-

plications des transformations autorisent une forme de modélisation du temps. Nous avons en particulier développé une stratégie fondée sur l'algorithme de D.T. Gillespie [Gil77] utilisé pour la simulation stochastique exacte de réactions chimiques mais correspondant plus généralement à des modèles à évènements discrets probabilisés.

La figure 1 présente quelques exemples d'applications réalisées en MGS. Toutes ces images ont été générées par l'interprète que j'ai développé. En particulier, l'image en bas à droite correspond à une simulation réalisée en collaboration avec le CIRAD et l'INRA où MGS a été utilisé pour simuler leur modèle de la croissance du méristème apical caulinaire d'*Arabidopsis thaliana*, couplant des transports d'hormones avec la migration et la différenciation cellulaires.

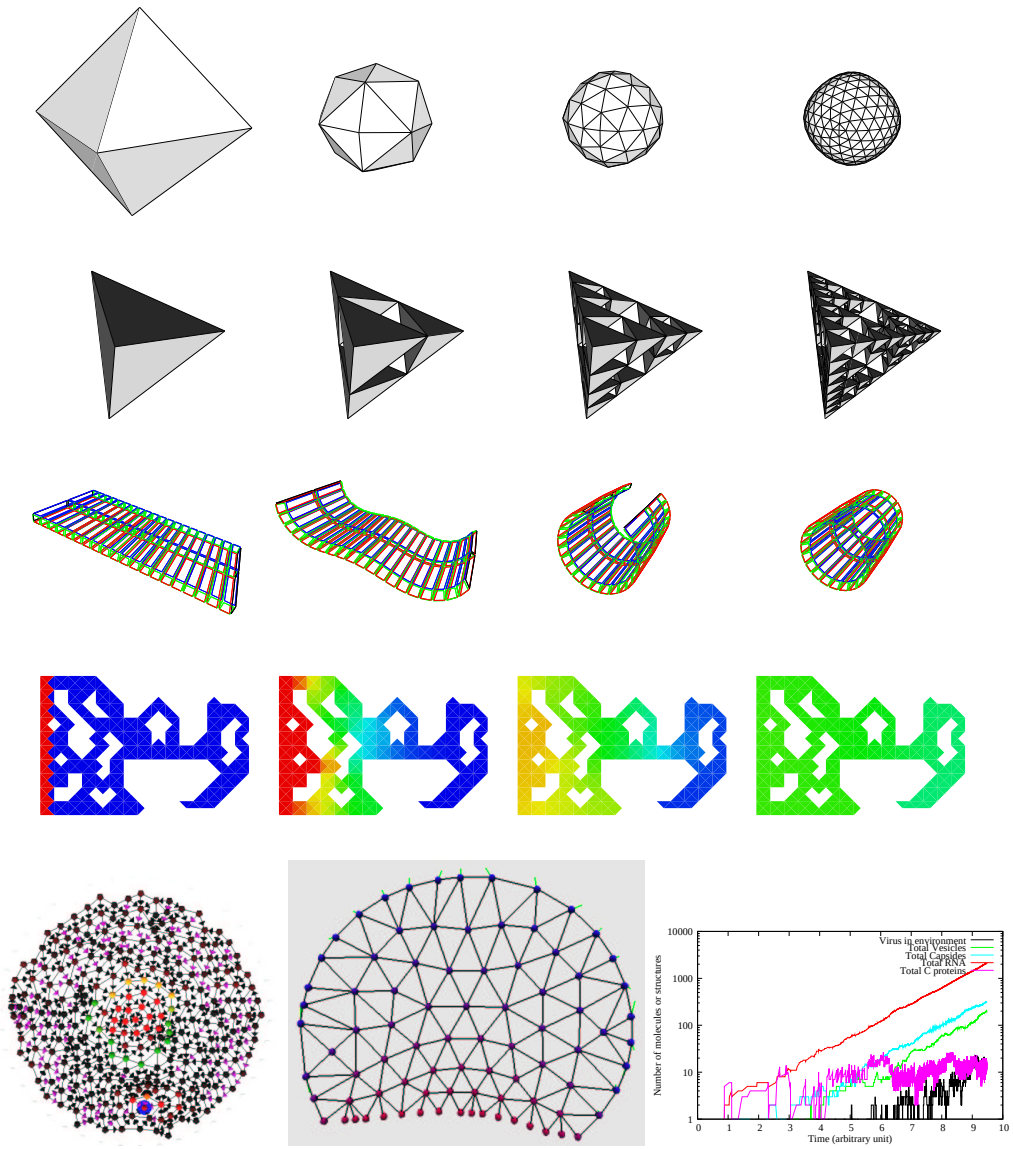

FIG. 1 – Quelques exemples de simulations réalisées en MGS : on trouve sur les 2 premières lignes, 2 exemples d'opérations de CAO implantées de façon déclarative : le raffinement de maillage (algorithme de Loop) et la génération d'une figure fractale (éponge de Sierpinski). La ligne 3 présente une modélisation du processus de neurulation, avec une modification topologique de la structure. Les images de la quatrième ligne correspondent à la simulation numérique en physique discrète de la diffusion de particules dans un fluide. Finalement sur la cinquième ligne, on trouve de gauche à droite l'illustration des flots d'auxine lors de la croissance du méristème chez *Arabidopsis thaliana*, la modélisation du déplacement cellulaire du spermatozoïde d'*Ascaris suum* sur un maillage adaptatif et une simulation stochastique de l'infection de virus de la forêt de Semliki.

Références

[BFL01] Jean-Pierre Banâtre, Pascal Fradet, and Daniel Le Métayer. Gamma and the chemical reaction model: Fifteen years after. *Lecture Notes in Computer Science*, 2235:17–44, 2001.

[BL86] Jean-Pierre Banâtre and Daniel Le Métayer. A new computational model and its discipline of programming. Technical Report RR-0566, INRIA, 1986.

[ES79] Manfred Eigen and Peter Schuster. *The Hypercycle: A Principle of Natural Self-Organization.* Springer, 1979.

[FB94] Walter Fontana and Leo W. Buss. "the arrival of the fittest": Toward a theory of biological organization. *Bulletin of Mathematical Biology*, 1994.

[GGMP02] Jean-Louis Giavitto, Christophe Godin, Olivier Michel, and Przemyslaw Prusinkiewicz. *Modelling and Simulation of biological processes in the context of genomics*, chapter "Computational Models for Integrative and Developmental Biology". Hermes, July 2002. Also republished as an high-level course in the proceedings of the Dieppe spring school on "Modelling and simulation of biological processes in the context of genomics", 12-17 may 2003, Dieppes, France.

[Gil77] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[GM01a] Jean-Louis Giavitto and Olivier Michel. MGS : a rule-based programming language for complex objects and collections. In Mark van den Brand and Rakesh Verma, editors, *Electronic Notes in Theoretical Computer Science*, volume 59. Elsevier Science Publishers, 2001.

[GM01b] Jean-Louis Giavitto and Olivier Michel. `MGS`: a programming language for the transformations of topological collections. Technical Report 61-2001, LaMI – Université d'Évry Val d'Essonne, May 2001.

[HGH93] John Y. Hung, Weibing Gao, and James C. Hung. Variable structure control: A survey. *IEEE Transactions on Industrial Electronics*, 40(1):2–22, 1993.

[Itk76] Yevgeny Itkis. *Control Systems of Variable Structure.* Wiley, 1976.

[Jan00] Patrik Jansson. *Functional Polytypic Programming.* PhD thesis, Chalmers University, 2000.

[Kah87] Gilles Kahn. Natural semantics. Technical Report 601, INRIA, February 1987.

[MSR91] Eric Mjolsness, David H. Sharp, and John Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152(4):429–454, 1991.

[Mun84] James Munkres. *Elements of Algebraic Topology.* Addison-Wesley, 1984.

[Pău00] Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 1(61):108–143, 2000.

[Pău01] Gheorghe Păun. From cells to computers: computing with membranes (P systems). *Biosystems*, 59(3):139–158, March 2001.

[PLH+90] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, Jim Hanan, et al. *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[Rau03] Erik Rauch. Discrete, amorphous physical models. *International Journal of Theoretical Physics*, 42(2):329–348, Februray 2003.

[RS92] Grzegorz Rozenberg and Arto Salomaa. *Lindenmayer Systems.* Springer, Berlin, 1992.

[Var79] Fransisco J. Varela. *Principle of Biological Autonomy.* McGraw-Hill/Appleton & Lange, 1979.

[vN66] John von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, 1966.