Transformations de graphes décorés

Pascale Le Gall, Romain Pascual

Chapitre 4

Transformations de graphes décorés

Pascale Le Gall
Romain Pascual

Les transformations de graphes sont une généralisation de la réécriture à des structures non linéaires, plébiscitées pour étudier l'évolution de systèmes complexes. L'approche usuelle par « double somme amalgamée » explore des règles définies par trois graphes décrivant les motifs supprimés, ajoutés et préservés lors de la transformation. Pour représenter des classes de données particulières, on ajoute de l'information sur les nœuds et arcs des graphes. En fonction des propriétés de l'information ajoutée, différentes représentations ou décorations [1] sont possibles. Dans ce chapitre, nous présenterons les règles transformations de graphes et explorerons différentes méthodes pour les étendre aux graphes décorés. Nous illustrerons ces notions à l'aide d'opérations de modélisation géométrique sur les cartes généralisées.

4.1 Introduction

L'étude des transformations de graphes comme l'application successive de règles de la forme

$$Initial\ Graph \rightarrow Modified\ Graph$$

1. Le terme « *décoré* » est une traduction du terme générique « decorated graphs » utilisé notamment dans [14] pour ajouter des informations aux graphes dans un contexte catégoriel.

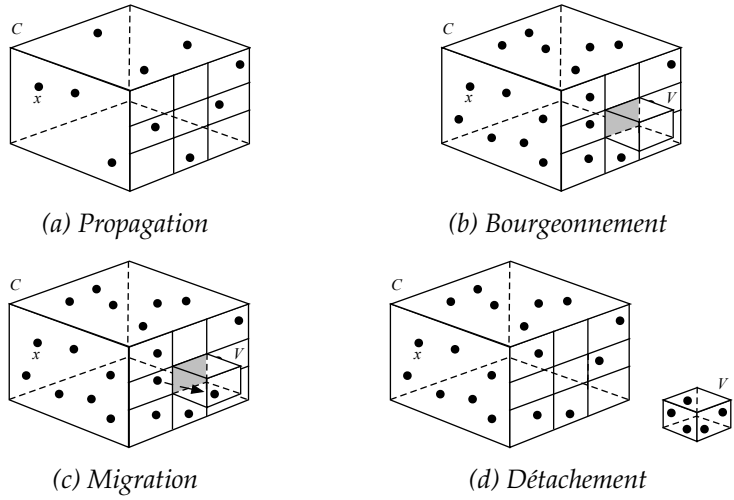


FIGURE 4.1 – *Modélisation de la création d'une vésicule.*

a été motivée par l'ambition de transposer le savoir-faire en systèmes de réécriture appliqués aux mots, termes ou formules dans le contexte des graphes et par la volonté de disposer d'un langage simple et haut-niveau pour décrire des modifications considérées comme standards dans les domaines d'application utilisant des graphes (e.g., refactorisation de code, transformation de modèles…). Quant à nous, notre intérêt pour les transformations de graphes est né de l'étude spatio-temporelle de l'appareil de Golgi. Il s'agit d'un organite cellulaire qui récupère les protéines fabriquées dans le réticulum endoplasmique d'une cellule, les fait maturer avant de les excréter vers le cytoplasme à l'aide de vésicules. L'appareil de Golgi se présente sous la forme d'une pile de saccules et différentes hypothèses existent quant aux mécanismes spatio-temporels en jeu.

Les figures 4.1a à 4.1d sont extraites de nos premiers travaux mettant en jeu les transformations de graphes et la modélisation géométrique avec l'objectif d'élaborer une simulation du fonctionnement de l'appareil de Golgi [26]. Elles illustrent de façon très simplifiée le processus biologique du bourgeonnement des vésicules qui encapsulent et excrètent des entités (protéines ou toxines) des compartiments subcellulaires. Dans un premier temps, une protéine x se propage dans un compartiment subcellulaire de type C (voir figure 4.1a). Si la concentration de x est suffisamment élevée, une vésicule de type V bourgeonne hors du compartiment (voir figure 4.1b). La face grise y représente une membrane factice entre le compartiment et la vésicule et préfigure la création d'une vésicule résultant du gonflement local d'un compartiment. Le transport de x du compartiment principal

vers la vésicule se fait en traversant cette face grisée (voir figure 4.1c). A la dernière étape (voir figure 4.1d), lorsque la vésicule est remplie de x, elle est déconnectée du compartiment principal. Ce processus peut être décrit comme l'application de 3 étapes successives : l'extrusion à partir d'une facette (de la figure 4.1a vers la figure 4.1b), le franchissement des surfaces par des protéines (de la figure 4.1b vers la figure 4.1c) et la séparation de deux volumes (de la figure 4.1c vers la figure 4.1d). Dès lors que les schémas de la figure 4.1 sont représentés par des graphes, ces trois étapes - extrusion, franchissement et séparation - sont susceptibles d'être modélisées par des règles de transformations de graphes. Si l'on encode les composantes topologiques (volumes, surfaces, sommets…) par une structure de graphe, ces règles modifient la structure des graphes. Si l'on encode les composantes géométriques (positions des sommets, concentration des protéines, type des compartiments, perméabilité des surfaces…) par des annotations attachées aux éléments du graphe (nœuds ou arcs), les transformations en jeu modifient les décorations des graphes en jeu. Ces décorations peuvent être de simples étiquettes (e.g. le type des compartiments), ou des valeurs d'un type de données sur lesquelles il est possible d'effectuer des calculs (concentration des protéines). Elles peuvent aussi être corrélées entre elles (perméabilité des membranes en lien avec la nature des compartiments). Il est alors essentiel que les transformations de graphes non seulement prennent en compte les décorations, mais aussi préservent leurs propriétés. Nous ne reviendrons pas dans ce chapitre sur les applications bio-chimiques, qui nécessiteraient beaucoup d'éléments de contextualisation pour être développées. Un lecteur intéressé peut consulter [25] pour la modélisation de l'appareil de Golgi ou [3] pour la modélisation de réactions bio-chimiques régies par des compartiments. À travers cette présentation succincte d'un domaine applicatif, nous avons cherché à suggérer l'importance de porter attention à la façon dont les décorations sur les graphes sont prises en charge par les transformations de graphes.

La structure et les illustrations de ce chapitre reprennent de nombreux éléments de la thèse [23]. Il est organisé de la sorte : la section 4.2 regroupe les principales références bibliographiques utilisées ; dans la section 4.3, nous introduisons quelques éléments de théorie des catégories ; dans la section 4.4, nous décrivons différentes classes de graphes en termes catégoriels ; dans la section 4.5, nous présentons l'approche, dite DPO, pour les transformations de graphes ; pour conclure, nous revenons rapidement en section 4.6 sur l'emploi des transformations de graphes dans le contexte de la modélisation géométrique.

4.2 Éléments bibliographiques

Le présent chapitre est une simple introduction au thème des transformations de graphes décorés. Nous en présenterons les notions clés, en laissant de côté les plus avancées ou confidentielles. Les transformations de graphes généralisent la réécriture des mots ou des termes aux graphes et sont aujourd'hui principalement introduites par le biais des catégories qui permettent de les définir à l'aide d'une construction abstraite et générique. Ce parti-pris de généralité permet de modifier différentes structures de graphes proches, en particulier portant différents types de décorations, selon une même démarche. Nous donnons ici quelques pointeurs vers deux types de références bibliographiques : les premières concernent les références qui ont établi certains résultats fondamentaux présentés dans ce document, et les secondes sont des ouvrages introductifs ou des notes de cours permettant d'approfondir certaines notions.

Théorie des catégories Les transformations de graphes constituent un domaine d'application de la théorie des catégories. Cette branche des mathématiques est née des travaux de S. Eilenberg et S. Mac Lane dans les années 40 et développée notamment par A. Grothendieck dans les années 60 et 70. La monographie de S. Mac Lane [22] reste la référence incontournable sur le sujet. Nous mentionnons aussi les ouvrages de M. Barr et C. Wells [1] pour sa vision plus informatique et de T. Leinster [19], plus accessible à un lecteur disposant d'un bagage mathématique relativement modeste.

Transformations de graphes Les catégories qui nous intéresseront dans ce chapitre sont celles décrivant les graphes. Les transformations de graphes et l'étude des grammaires de graphes datent des années 60. La série de trois livres [27, 11, 10] publiés à la fin des années 90 dresse un panorama très détaillé du domaine. Cependant, [8] demeure probablement l'ouvrage de référence pour une compréhension détaillée des constructions théoriques sous-jacentes aux transformations de graphes. Les tutoriels et articles introductifs [7, 9] de H. Ehrig permettront à un lecteur impatient de comprendre rapidement les constructions et motivations. Plus récemment, le tutorial de B. König et al. [17] reprend des constructions plus anciennes pour présenter les transformations de graphes attribués sans utiliser explicitement les notions sous-jacentes liées aux catégories. De même, le livre de R. Heckel et G. Taentzer [14] présente une vision moderne des transformations de graphes à destination de non-experts du domaine. Nous présentons ici

l'approche par double somme amalgamée des transformations de graphes mais tenons à préciser que d'autres approches existent (SPO [21], SqPO [5] ou encore PBPO [4]) pour lesquelles il convient essentiellement de changer les constructions universelles sous-jacentes à la réécriture. Enfin, nous tenons à mentionner que la notion d'adhésivité utilisée ici a été introduite par S. Lack et P. Sobociński [18] et englobe notamment les topos et les préfaisceaux.

Transformations des graphes pour la modélisation géométrique Les domaines d'application des transformations de graphes sont multiples : à titre d'exemples, on peut citer l'ingénierie logicielle (ingénierie des modèles UML, retro-ingénierie de code) ou les langages à base de règles pour les interactions chimiques. Dans ce chapitre, nous privilégions l'emploi des modèles combinatoires (en particulier, celui des cartes généralisées) pour la représentation des objets nD. Les premières définitions des modèles combinatoires ont été de nature algébrique, sous la forme d'ensembles munis d'une permutation par dimension [20]. Dès lors que les modèles combinatoires sont définis comme des classes de graphes particuliers, il devient possible de considérer une opération de modélisation géométrique (triangulation, extrusion, chanfreinage, etc) comme une règle de transformations de graphes. Dans le cadre de l'outil Jerboa [2], les modèles utilisés sont les cartes généralisées et les transformations de graphes sont définies par une double somme amalgamée. Dans le langage MGS [13], il est possible de manipuler des complexes cellulaires abstraits, offrant ainsi un langage pour concevoir des opérations de modélisation géométrique : une construction du triangle de Sierpinski et de l'éponge de Menger est présentée dans [28]. De manière plus éloignée, les transformations de graphes ont aussi permis la reconstruction de scènes d'intérieur [16], de tunnels [29] ou la topologie d'un réseau de routes pour la génération de villes [12].

4.3 Éléments de théorie des catégories

4.3.1 Catégories, objets et morphismes

Définition 4.3.1 (Catégorie)**.** *Une catégorie $\mathscr{A}$ consiste en :*

- *une collection $\mathcal{O}(\mathscr{A})$ d'objets ;*
- *pour tout $A, B \in \mathcal{O}(\mathscr{A})$, une collection $\mathscr{A}(A, B)$ de morphismes de A vers B.*

— *pour tout* $A, B, C \in \mathcal{O}(\mathscr{A})$*, une fonction*

$$\begin{cases} \mathscr{A}(B,C) \times \mathscr{A}(A,B) & \to & \mathscr{A}(A,C) \\ (g,f) & \mapsto & g \circ f \end{cases}$$

appelée composition, satisfaisant la loi d'associativité suivante : pour tout $f \in \mathscr{A}(A,B)$*,* $g \in \mathscr{A}(B,C)$ *et* $h \in \mathscr{A}(C,D)$*,* $(h \circ g) \circ f = h \circ (g \circ f)$*;*

— *pour tout* $A \in \mathcal{O}(\mathscr{A})$*, il existe un élément* 1_A *de* $\mathscr{A}(A,A)$*, appelé l'identité sur* A*, tel que pour tout* $f \in \mathscr{A}(A,B)$*,* $f \circ 1_A = f = 1_B \circ f$*.*

La définition d'une catégorie utilise la notion informelle de *collection*, il s'agit d'une manière de caractériser un « tas de choses » sans se soucier de savoir si ces choses forment un ensemble, une classe, ou une notion formelle quelconque liée à un type. Par simplicité, on note aussi $A \in \mathscr{A}$ pour $A \in \mathcal{O}(\mathscr{A})$ et $f \colon A \to B$ ou $A \xrightarrow{f} B$ pour $f \in \mathscr{A}(A,B)$. Suivant les auteurs, la collection des morphismes de A vers B est aussi noté $\mathrm{Hom}_{\mathscr{A}}(A,B)$ ou $\mathrm{Mor}_{\mathscr{A}}(A,B)$.

Exemple 4.3.2 (Catégories particulières)**.**

— *La catégorie vide* $\emptyset$ *est la catégorie ne contenant aucun objet et aucun morphisme.*

— *La catégorie triviale* $*$ *est la catégorie contenant un seul objet* $*$ *et le seul morphisme identité* 1_* *associé à cet élément.*

— *La catégorie* **Set** *a pour objets, les ensembles et pour morphismes, les fonctions entre les ensembles. En particulier, les morphismes identités sont les fonctions identités associées à chaque ensemble, et la composition des morphismes est la composition des fonctions.*

— *Plus généralement, de nombreuses structures mathématiques (préordres, monoïdes, groupes, anneaux, espaces topologiques) peuvent être munies d'une structure de catégorie.*

Certains objets jouent un rôle particulier dans une catégorie, à condition d'exister.

Définition 4.3.3 (Objets initiaux et terminaux)**.** *Un objet* $\varnothing_{\mathscr{A}}$ *d'une catégorie* $\mathscr{A}$ *est dit initial si pour tout objet* $A \in \mathscr{A}$*, il existe un unique morphisme* $\varnothing_{\mathscr{A}} \to A$*. Par dualité, un objet* $\mathtt{1}_{\mathscr{A}}$ *d'une catégorie* $\mathscr{A}$ *est dit terminal si pour tout objet* $A \in \mathscr{A}$*, il existe un unique morphisme* $A \to \mathtt{1}_{\mathscr{A}}$*.*

Les objets initiaux d'une catégorie $\mathscr{A}$, lorsqu'ils existent, sont uniques à isomorphisme près. On parle alors de l'objet initial de $\mathscr{A}$ en le notant $\varnothing_{\mathscr{A}}$. Par dualité, les objets terminaux sont aussi uniques à isomorphisme près, sous réserve qu'ils existent. On note alors $!_A$ l'unique morphisme $A \to \mathtt{1}_{\mathscr{A}}$, pour A objet de $\mathscr{A}$.

Exemple 4.3.4.
- *L'unique objet de la catégorie triviale $*$ est à la fois initial et terminal.*
- *Dans* **Set**, *l'ensemble vide $\emptyset$ est initial et tous les singletons sont terminaux.*

Les notions d'isomorphismes, de monomorphismes et d'épimorphismes généralisent dans le contexte des catégories les notions de fonctions bijectives, injectives et surjectives pour les ensembles.

Définition 4.3.5 (Isomorphismes, monomorphismes, épimorphismes)**.** *Soit $\mathscr{A}$ une catégorie.*
- *Un morphisme $f : A \to B$ dans $\mathscr{A}$ est un isomorphisme s'il existe un morphisme $g : B \to A$ dans $\mathscr{A}$ vérifiant $g \circ f = 1_A$ et $f \circ g = 1_B$. g est alors appelé l'inverse de f et A et B sont dits* isomorphes.
- *Un morphisme $f : A \to B$ dans $\mathscr{A}$ est un monomorphisme ou simplement un mono, $f : A \hookrightarrow B$, s'il est simplifiable à gauche i.e., pour tout objet $X \in \mathscr{A}$ et pour toute paire de morphismes $g, g' : X \to A$, si $f \circ g = f \circ g'$ alors $g = g'$.*
- *Un morphisme $f : A \to B$ dans $\mathscr{A}$ est un épimorphisme, ou simplement un épi, s'il est simplifiable à droite i.e., pour tout objet $X \in \mathscr{A}$ et pour toute paire de morphismes $g, g' : B \to X$, si $g \circ f = g' \circ f$ alors $g = g'$.*

Dans les catégories qui nous intéressent pour ce chapitre, il est possible d'interpréter un monomorphisme $f : A \hookrightarrow B$ comme l'identification une sous-structure de l'objet cible B isomorphe à l'objet source A.

4.3.2 Diagrammes commutatifs, produits et sommes

De nombreuses propriétés des catégories sont décrites à l'aide de diagrammes commutatifs. Un diagramme exprime des équations entre morphismes de mêmes source et cible. Par exemple, le diagrame

$$\begin{array}{ccc} B & \xleftarrow{f} & A \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle i} \\ C & \xrightarrow[h]{} & D \end{array}$$

exprime l'égalité $h \circ g \circ f = i$. Plus généralement, un diagramme est dit *commutatif* si tous les couples de chemins de même source et même cible définissent deux morphismes égaux par composition des morphismes le long des chemins. Parmi les diagrammes remarquables, on distingue les

carrés se présentant sous la forme :

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle f'}\downarrow & & \downarrow{\scriptstyle g} \\ C & \xrightarrow[g']{} & D \end{array}$$

pour exprimer l'égalité $g \circ f = g' \circ f'$ ainsi que les diagrammes de la forme $A \xleftarrow{g} C \xrightarrow{h} B$ (resp. $A \xrightarrow{g} C \xleftarrow{h} B$) appelés *spans* (resp. *cospans*). Ces derniers sont en quelque sorte incomplets car ils représentent des amorces de diagrammes commutatifs.

La théorie des catégories a recours à des constructions dites « universelles », qui cherchent d'une certaine manière à identifier la structure la plus générale vis-à-vis d'un diagramme. Par exemple, le produit de deux objets peut se comprendre comme le span le plus général construit sur ces deux objets.

Définition 4.3.6 (Produit)**.** *Soient A et B deux objets d'une catégorie $\mathscr{A}$. Le produit de A et B est la donnée d'un span $A \xleftarrow{p_A} P \xrightarrow{p_B} B$ dans $\mathscr{A}$ vérifiant que pour tout autre span de la forme $A \xleftarrow{f_A} X \xrightarrow{f_B} B$ dans $\mathscr{A}$, il existe un unique morphisme $f : X \to P$ assurant que le diagramme ci-dessous commute :*

$$\begin{array}{ccccc} & & X & & \\ & {\scriptstyle f_A}\swarrow & \vdots\,{\scriptstyle \exists! f} & \searrow{\scriptstyle f_B} & \\ A & \xleftarrow[p_A]{} & P & \xrightarrow[p_B]{} & B \end{array}$$

Les morphismes p_A and p_B sont appelés projections.

Ici, la notion de « span le plus général » est encodé par l'existence et l'unicité du morphisme $f : X \to P$, notée dans le diagramme de la façon suivante : $A \xrightarrow{\exists! f} B$. De plus, cette construction permet de caractériser *le* produit entre deux objets à isomorphisme près : si A et B admettent deux produits $A \xleftarrow{p_A} P \xrightarrow{p_B} B$ et $A \xleftarrow{p'_A} P' \xrightarrow{p'_B} B$, la propriété du produit induit un isomorphisme entre P et P'. Par abus, le produit de A et B sera simplement décrit par l'objet P au centre du span, dénoté par $A \times B$, laissant implicites les projections p_A et p_B.

Exemple 4.3.7. *Le produit de deux ensembles A et B dans* **Set** *est le produit cartésien $A \times B$ usuel muni des projections sur chacune des composantes.*

Les produits fibrés (pullbacks) sont des produits pour lesquels les deux objets partagent des morphismes vers un troisième objet.

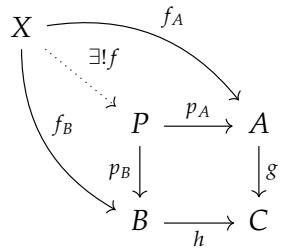
Définition 4.3.8 (Produit fibré)**.** *Soit $A \xrightarrow{g} C \xleftarrow{h} B$ un cospan dans une catégorie $\mathscr{A}$. Le produit fibré de $A \xrightarrow{g} C \xleftarrow{h} B$ est un span $A \xleftarrow{p_A} P \xrightarrow{p_B} B$ vérifiant que le carré*

$$\begin{array}{ccc} P & \xrightarrow{p_A} & A \\ {\scriptstyle p_B}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[h]{} & C \end{array} \tag{4.1}$$

est commutatif et que pour tout carré commutatif

$$\begin{array}{ccc} X & \xrightarrow{f_A} & A \\ {\scriptstyle f_B}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[h]{} & C \end{array}$$

dans $\mathscr{A}$, il existe un unique morphisme $f : X \to P$ assurant que le diagramme



est commutatif.

De manière identique, le produit fibré est unique à isomorphisme près, s'il existe. Dans les catégories construites à partir des ensembles, les produits fibrés admettent souvent une construction ensembliste.

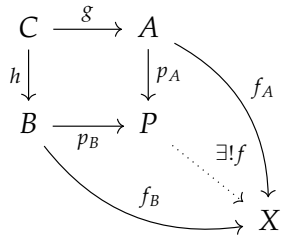
Exemple 4.3.9. *Le produit fibré de $A \xrightarrow{g} C \xleftarrow{h} B$ dans* **Set** *est l'ensemble $P = \{(a, b) \in A \times B \mid g(a) = h(b)\}$ muni des projections $p_A(a, b) = a$ et $p_B(a, b) = b$.*

En théorie des catégories, il est possible d'obtenir de nouvelles constructions par *dualité*, c'est-à-dire en inversant le sens des morphismes. Ainsi la notion duale de produit fibré est la somme amalgamée.

Définition 4.3.10 (Somme amalgamée)**.** *La somme amalgamée (pushout) d'un span $A \xleftarrow{g} C \xrightarrow{h} B$ dans une catégorie $\mathscr{A}$ est le cospan $A \xrightarrow{p_A} P \xleftarrow{p_B} B$ vérifiant que le carré*

$$\begin{array}{ccc} C & \xrightarrow{g} & A \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle p_A} \\ B & \xrightarrow[p_B]{} & P \end{array}$$

commute et est universel avec cette propriété. Autrement dit pour tout autre carré commutatif, il existe un unique morphisme tel que le diagramme suivant est commutatif :



Les morphismes p_A et p_B sont appelés coprojections de la somme amalgamée.

Exemple 4.3.11. *La somme amalgamée $A \xleftarrow{g} C \xrightarrow{h} B$ dans* **Set** *est l'ensemble $P = (A \sqcup B)/\sim$, où $\sqcup$ est l'union disjointe*[2] *de deux ensembles et $\sim$ est la relation d'équivalence définie sur C par $g(c) \sim h(c)$ pour tout $c \in C$. La coprojection $p_A \colon A \to P$ (resp. $p_B \colon B \to P$) envoie un élément $a \in A$ (resp. $b \in B$) sur sa classe d'équivalence dans P.*

4.3.3 Foncteurs et constructions de catégories

Les morphismes entre catégories sont appelés foncteurs :

Définition 4.3.12 (Foncteur)**.** *Un foncteur $F \colon \mathscr{A} \to \mathscr{B}$ entre deux catégories est défini par :*

- *une fonction $F : \mathcal{O}(\mathscr{A}) \to \mathcal{O}(\mathscr{B})$,*
- *et pour tout couple $A, A' \in \mathscr{A}$, une fonction $F : \mathscr{A}(A, A') \to \mathscr{B}(F(A), F(A'))$,*

satisfaisant les propriétés suivantes :

- *$F(g \circ f) = F(g) \circ F(f)$ pour tous les morphismes $f \colon A \to B$ et $g \colon B \to C$ dans $\mathscr{A}$;*

2. Pour A et B deux ensembles, $A \sqcup B$ est un ensemble isomorphe à l'ensemble $(\{0\} \times A) \cup (\{1\} \times B)$.

— $F(1_A) = 1_{F(A)}$ *pour tout objet* $A \in \mathscr{A}$.

Intuitivement, la construction d'un foncteur assure que pour toute séquence de morphismes $A_0 \xrightarrow{f_1} A_1 \xrightarrow{f_2} \dots \xrightarrow{f_n} A_n$ (avec $n \geqslant 0$) dans $\mathscr{A}$, il existe un unique morphisme $F(A_0) \to F(A_n)$ dans $\mathscr{B}$. Les foncteurs constituent les morphismes de la catégorie **Cat** dont les objets sont les petites [3] catégories.

Définition 4.3.13 (Catégorie slice)**.** *Soit A un objet d'une catégorie $\mathscr{A}$. La catégorie slice [4] de $\mathscr{A}$ sur A, notée $\mathscr{A}/A$, est définie par :*

— *les objets de $\mathcal{O}(\mathscr{A}/A)$ sont les morphismes $X \to A$ dans $\mathscr{A}$,*

— *pour deux objets $f\colon X \to A$ et $f'\colon X' \to A$ dans $\mathscr{A}/A$, l'ensemble $\mathscr{A}/A(f, f')$ des morphismes de f vers f' sont les morphismes $g\colon X \to X'$ assurant le diagramme commutatif suivant :*

$$\begin{array}{ccc} X & \xrightarrow{\quad g \quad} & X' \\ & {\scriptstyle f}\searrow \quad \swarrow{\scriptstyle f'} & \\ & A & \end{array}$$

On remarquera par exemple que le morphisme 1_A est terminal dans $\mathscr{A}/A$.

Définition 4.3.14 (Catégorie comma)**.** *Soit $\mathscr{A} \xrightarrow{P} \mathscr{C} \xleftarrow{Q} \mathscr{B}$ un cospan dans* **Cat**. *La catégorie comma $(P \downarrow Q)$ est définie par :*

— *les objets de $(P \downarrow Q)$ sont les triplets de la forme (A, h, B) avec $A \in \mathscr{A}$, $B \in \mathscr{B}$, et $h\colon P(A) \to Q(B)$ morphisme de $\mathscr{C}$;*

— *les morphismes $(A, h, B) \to (A', h', B')$ sont définis par des couples de la forme $(f\colon A \to A',\ g\colon B \to B')$ assurant le carré commutatif suivant :*

$$\begin{array}{ccc} P(A) & \xrightarrow{P(f)} & P(A') \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h'} \\ Q(B) & \xrightarrow[Q(g)]{} & Q(B') \end{array}$$

Les catégories slices sont des cas particuliers de catégories comma. En effet, la catégorie slice $\mathscr{A}/A$ est la catégorie comma $(1_{\mathscr{A}} \downarrow A)$, où $1_{\mathscr{A}} : \mathscr{A} \to \mathscr{A}$ est le foncteur identité sur $\mathscr{A}$ et l'objet A est identifié au foncteur $A : * \to \mathscr{A}$ qui envoie l'unique objet de la catégorie $*$ sur A.

3. Une catégorie est dite petite si ses collections d'objets et de morphismes sont des ensembles.

4. Formulation anglaise, en français on trouve parfois la notion de catégorie des objets de $\mathscr{A}$ au dessous de A.

4.4 Catégories à base de graphes

Il y a autant de catégories de graphes qu'il y a de classes de graphes que l'on peut souhaiter particulariser : arcs orientés ou non, nœuds et arcs étiquetés ou non, arcs reliant deux nœuds ou plus, etc.

4.4.1 Catégorie des multi-graphes

La première catégorie des graphes a pour objets, des ensembles de nœuds reliés par des arcs entre deux nœuds. Cette construction qui reste la plus générique autorise les arcs parallèles et les boucles. Par défaut c'est à cette construction que nous nous référerons lorsque nous parlerons de graphe. La source et la cible de chaque arc dans un graphe sont identifiées par deux fonctions s (pour la source) et t (pour la cible), de sorte qu'un graphe peut être décrit par le diagramme suivant [5] dans **Set** :

$$E \underset{t}{\overset{s}{\rightrightarrows}} V \tag{4.2}$$

où V est l'ensemble des nœuds et E est l'ensemble des arcs. En l'absence de contraintes sur s et t, les graphes peuvent contenir des arcs parallèles ou des boucles. Ainsi, la classe des multigraphes, ou graphes par abus, admet n'importe quelle configuration pour ses arcs alors que la classe des graphes simples autorise au plus un arc entre deux nœuds.

Définition 4.4.1 (Graphe)**.** *Un graphe $G = (V_G, E_G, s_G, t_G)$ est défini par un ensemble de nœuds V_G, un ensemble d'arcs E_G et deux fonctions $s_G, t_G : E_G \to V_G$.*

La définition d'un graphe ne présuppose pas que les ensembles de nœuds et d'arcs soient finis. Toutes les constructions présentées dans ce chapitre restent valides dans le cas infini bien que nous ne présentons que des exemples finis. Par ailleurs, les indices $_G$ des composantes d'un graphe seront omis en l'absence d'ambiguïté.

Définition 4.4.2 (Morphisme de graphes)**.** *Un morphisme de graphes $m = (m_V, m_E) : G \to H$ est défini par deux fonctions $m_V : V_G \to V_H$ et $m_E : E_G \to E_H$ préservant les sources et cibles des arcs, i.e., assurant les diagrammes commutatifs suivants :*

5. Un lecteur familier avec la théorie des catégories remarquera qu'il faut en réalité renverser les flèches du diagramme pour obtenir la construction par préfaisceau de **Graph**.

$$\begin{array}{ccc} E_G & \xrightarrow{s_G} & V_G \\ {\scriptstyle m_E}\downarrow & & \downarrow{\scriptstyle m_V} \\ E_H & \xrightarrow[s_H]{} & V_H \end{array} \qquad \begin{array}{ccc} E_G & \xrightarrow{t_G} & V_G \\ {\scriptstyle m_E}\downarrow & & \downarrow{\scriptstyle m_V} \\ E_H & \xrightarrow[t_H]{} & V_H \end{array} \tag{4.3}$$

La catégorie des graphes **Graph** *a pour objets les graphes et pour morphismes, les morphismes de graphes.*

Les indices V et E pourront être omis pour écrire $m(x)$ que x soit un nœud ou un arc de G. Dans **Graph**, le graphe vide $G_\emptyset = (\emptyset, \emptyset, 1_\emptyset, 1_\emptyset)$ est initial et le graphe réduit à un seul nœud avec une boucle $(\{v\}, \{e\}, s\colon e \mapsto v, t\colon e \mapsto v)$ est terminal.

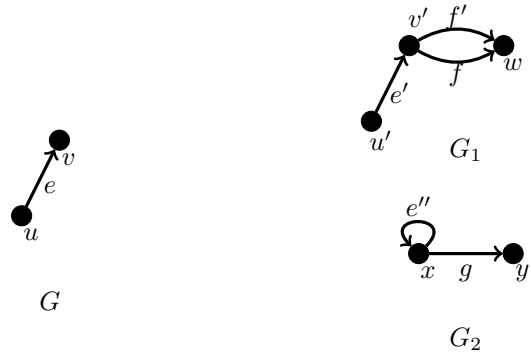


FIGURE 4.2 – *Exemples de graphes*

Exemple 4.4.3. *La figure 4.2 présente trois graphes G, G_1 et G_2. Le graphe G possède 2 nœuds (u, v) et un arc (e). Le graphe G_1 possède 3 nœuds (u', v' et w) et 3 arcs (e', f et f'). Finalement, le graphe G_2 possède 2 nœuds (x et y) et 2 arcs (e'' et g). Pour chacun des arcs, les nœuds sources et cibles sont identifiés par l'orientation de la flèche représentant l'arc. Par exemple les source et cible de e sont respectivement u et v.*

Il est possible de définir des morphismes de G dans G_1 ou G_2 ou de G_1 dans G_2. Par exemple, on peut considérer le morphisme $(m_V, m_E) : G \to G_1$ défini par ses composantes $m_V\colon u \mapsto u', v \mapsto v'$ et $m_E\colon e \mapsto e'$ et $(n_V, n_E) : G_1 \to G_2$ défini par ses composantes $n_V\colon u' \mapsto x, v' \mapsto x, w \mapsto y$ et $n_E\colon e' \mapsto e'', f \mapsto g, f' \mapsto g$. Un lecteur attentif remarquera que ces deux morphismes préservent effectivement les sources et cibles des arcs.

Un morphisme $m = (m_V, m_E)\colon G \to H$ dans **Graph** est un mono (resp. un épi) si et seulement si m_V and m_E sont des fonctions injectives (resp. surjectives).

4.4.2 Catégorie des graphes non orientés

Il peut être préférable de manipuler des graphes non-orientés. Il est possible de définir des graphes de différentes manières, par exemple en équipant un graphe d'une involution sur les arcs associant à chaque arc son arc inverse, ou bien en remplaçant la construction sous-forme de paire source-cible d'un arc par une construction sous forme d'ensemble. C'est cette seconde construction que nous considérons ici. [6]

Un arc est non orienté, s'il possède un arc inverse. Un graphe non orienté est alors un graphe $G = (V, E, s, t)$ équipé d'une involution $i\colon E \to E$ vérifiant $s \circ i = t$ et $t \circ i = s$ ou bien peut être défini en remarquant qu'un arc est un ensemble de nœuds de taille 2 (ou 1 pour le cas des boucles) plutôt qu'une paire de nœuds, source et cible.

Définition 4.4.4 (Graphe non orienté)**.** *Soit $P_{(1,2)}\colon \mathbf{Set} \to \mathbf{Set}$ le foncteur qui associe à un ensemble S l'ensemble de ses sous-ensembles de cardinalité 1 ou 2. La catégorie des graphes non orientés* **uGraph** *a pour objets les graphes G définis par*

- *un ensemble de nœuds V_G,*
- *un ensemble d'arcs E_G,*
- *une fonction d'incidence $i_G\colon E_G \to P_{(1,2)}(V_G)$ qui associe à chaque arc l'ensemble de ses nœuds incidents,*

et pour morphismes, les morphismes $(m_V, m_E)\colon G \to H$ définis par $m_V\colon V_G \to V_H$ et $m_E\colon E_G \to E_H$ assurant que le carré ci-dessous commute :

$$\begin{array}{ccc} E_G & \xrightarrow{\quad m_E \quad} & E_H \\ {\scriptstyle i_G}\downarrow & & \downarrow{\scriptstyle i_H} \\ P_{(1,2)}(V_G) & \xrightarrow[P_{(1,2)}(m_V)]{} & P_{(1,2)}(V_H) \end{array}$$

Autrement dit, **uGraph** est la catégorie comma $(1_{\mathbf{Set}} \downarrow P_{(1,2)})$ avec $1_{\mathbf{Set}}\colon \mathbf{Set} \to \mathbf{Set}$ le foncteur identité sur **Set**.

Exemple 4.4.5. *La figure 4.3 présente une version non orientée des graphes de la figure 4.2. Les nœuds incidents d'un arc sont l'ensemble des nœuds aux extrémités de l'arc en question. Par exemple, $i_G(e) = \{u, v\}$ et $i_{G_2}(e'') = \{x\}$. Un exemple de morphismes entre les graphes non orientés G et G_2 est donné par $u \mapsto x, v \mapsto y$ et $e \mapsto g$.*

6. On remarquera qu'une telle approche permet de construire les hypergraphes en levant la condition de taille sur les arcs.

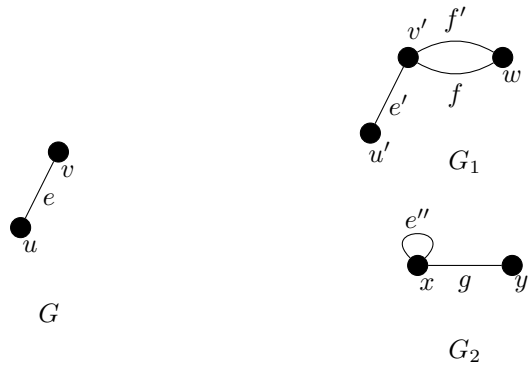


FIGURE 4.3 – *Exemples de graphes non orientés*

4.4.3 Catégorie des graphes étiquetés

Les graphes (ou les graphes non-orientés) représentent essentiellement des relations entre des éléments. Dans leur approche algébrique, les graphes sont étendus avec des décorations qui permettent de représenter l'ajout d'information sur la structure, que ce soit sur les éléments ou sur les relations. La première possibilité consiste à ajouter une information prise dans un ensemble fini de possibilité, typiquement un alphabet. On parle alors de graphe étiqueté ou coloré. Intuitivement, l'ajout des fonctions d'étiquetage, $l_E : E \to \Sigma_E$ pour les arcs et $l_V : V \to \Sigma_V$ pour les nœuds, peut être visualisé à l'aide du diagramme :

$$\Sigma_E \xleftarrow{le} E \underset{t}{\overset{s}{\rightrightarrows}} V \xrightarrow{lv} \Sigma_V$$

Définition 4.4.6 (Graphes étiquetés)**.** *Soit $\Sigma = (\Sigma_V, \Sigma_E)$ un couple d'alphabets. Un graphe étiqueté sur Σ est un tuple $G = (V_G, E_G, s_G, t_G, lv_G, le_G)$ où (V_G, E_G, s_G, t_G) est un graphe et où $lv_G \colon V_G \to \Sigma_V$ et $le_G \colon E_G \to \Sigma_E$ sont respectivement les fonctions d'étiquetage des nœuds et des arcs. Les morphismes des graphes étiquetés sont des morphismes de graphes préservant les étiquetage. Autrement dit, $m = (m_V, m_E) \colon G \to H$ est un morphisme de graphes étiqueté si les triangles ci-dessous commutent dans* **Set**.

$$\begin{array}{ccc} V_G & \xrightarrow{m_V} & V_H \\ & {\scriptstyle lv_G}\searrow \quad \swarrow{\scriptstyle lv_H} & \\ & \Sigma_V & \end{array} \qquad \begin{array}{ccc} E_G & \xrightarrow{m_E} & E_H \\ & {\scriptstyle le_G}\searrow \quad \swarrow{\scriptstyle le_H} & \\ & \Sigma_E & \end{array}$$

Les graphes étiquetés et leurs morphismes forment une catégorie notée Σ**-Graph**.

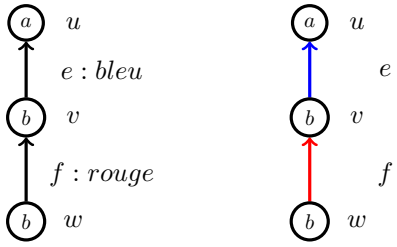


FIGURE 4.4 – *Graphes étiquetés*

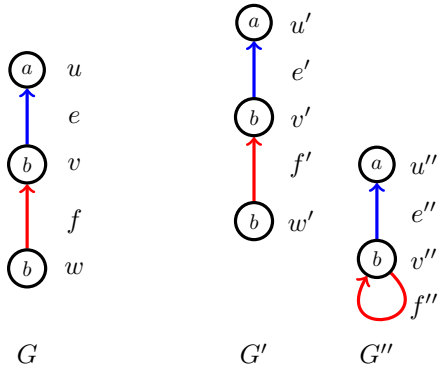


FIGURE 4.5 – *Morphismes de graphes étiquetés*

Exemple 4.4.7. *La figure 4.4 inclut deux visualisations d'un même graphe étiqueté pour lequel les nœuds sont étiquetés par une lettre de l'alphabet* $\mathbb{A} = \{a, b, \ldots, z\}$ *et les arcs par une couleur de l'ensemble* $\mathbb{C} = \{bleu, rouge, noir, vert, \ldots\}$. *Dans les deux versions du graphe, les étiquettes des nœuds sont placées dans le nœud lui-même. De façon générale, l'étiquette associée à un arc est positionnée à côté de son nom comme dans le graphe à gauche. En revanche, en tirant parti du choix des couleurs comme ensemble d'étiquettes pour les arcs, l'étiquette de l'arc est donnée par sa couleur dans le graphe de droite, ce qui permet une visualisation plus compacte du même graphe.*

Exemple 4.4.8. *La figure 4.5 contient trois graphes de* $(\mathbb{A}, \mathbb{C})$**-Graph**. *Un morphisme de graphes étiquetés de G vers G′ peut être défini en associant à chaque élément de G, nœud ou arc, sa version primée dans G′. De même, un morphisme de graphes étiquetés de G vers G″ peut être défini par les associations* $u \mapsto u'', v \mapsto v'', w \mapsto v''$ *pour les nœuds et* $e \mapsto e'', f \mapsto f''$ *pour les arcs.*

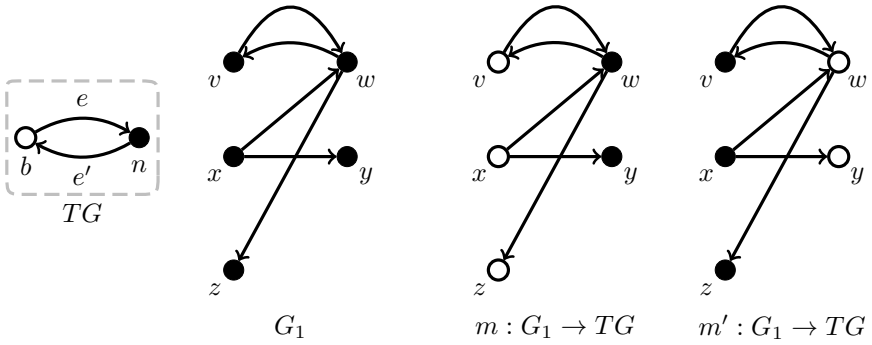


FIGURE 4.6 – *Graphe type pour les graphes bipartis*

4.4.4 Catégorie des graphes typés

L'ajout d'étiquettes sur un graphe permet d'incorporer des informations, de sorte que les éléments du graphes sont partitionnés selon leur étiquette. Cette idée peut être étendue en ajoutant des relations entre les étiquettes. On peut par exemple vouloir spécifier qu'un arc rouge ne peut exister qu'entre deux nœuds d'étiquette b. Ces relations entre les étiquettes seront encodées dans un graphe dédié TG, appelé graphe type. On peut alors typer un graphe G selon TG : les nœuds de TG représenteront les types de nœuds de G et les arcs de TG représenteront les types des arcs de G. La catégorie de tous les graphes typés par le graphe type TG est la catégorie **Graph**/ TG, obtenue par slicing.

Définition 4.4.9. *La catégorie* $\mathbf{Graph}_{\mathrm{TG}}$ *des graphes typés sur un graphe type* TG *est la catégorie slice* **Graph**/TG.

Notons que cette construction par slicing des graphes typés entraîne que l'on ne manipule plus à proprement parlé des graphes mais des morphismes vers un graphe type. En particulier, il peut exister plusieurs morphismes $G \to$ TG, chacun de ces morphismes décrit effectivement un graphe typé distinct. Néanmoins, une fois que le graphe type TG est donné, on représentera les graphes typés de manière analogue aux graphes étiquetés, c'est-à-dire en inscrivant les types sur les éléments du graphe G.

Exemple 4.4.10. *Le graphe type* TG *de la figure 4.6 permet de décrire la classe des graphes bipartis : il existe deux types de nœuds (b dessiné d'intérieur blanc et n d'intérieur noir) ; il est possible de considérer des arcs d'un type vers l'autre mais pas d'un type vers lui-même. Notons que les types e et e′ explicitent simplement le sens de passage d'un type à l'autre mais ne sont pas porteurs de plus d'information. Considérons le graphe* G_1 *sur cette même figure. On peut le typer de deux manières*

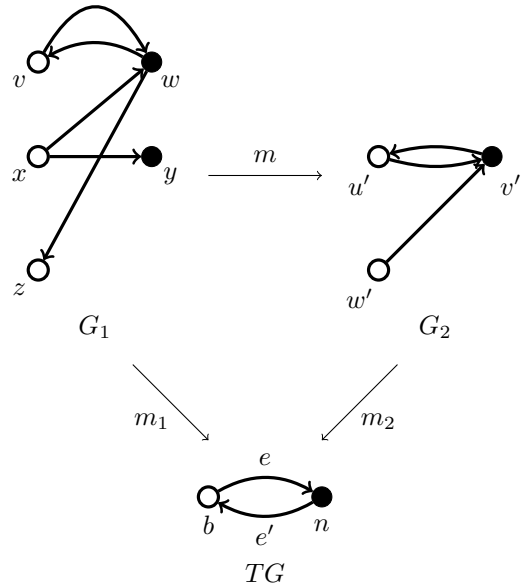


FIGURE 4.7 – *Morphismes de graphes bipartis*

par TG. *La première est donnée par le morphisme induit par la fonction qui associe les sommets v, x et z à b et les sommets w et y à n. On obtient alors le graphe typé* $m\colon G_1 \to \mathrm{TG}$. *En inversant b et n, on obtient le graphe typé* $m'\colon G_1 \to \mathrm{TG}$. *Notons que l'on representera un graphe typé par annotation de la source du morphisme afin d'en simplifier la visualisation, comme dans pour m et m′ dans la figure 4.6.*

La construction par slicing des graphes typés définit les morphismes des graphes typés comme des triangles commutatifs avec le graphe type TG comme cible. Cette construction assure que le type des nœuds ou des arcs ne sont pas modifiés par les morphismes des graphes typés.

Exemple 4.4.11. *La figure 4.7 illustre un morphisme de graphes bipartis. Plus précisément, le morphisme* $m\colon G_1 \to G_2$ *est défini par :*

- *pour les nœuds,* $v \mapsto u'$, $w \mapsto v'$, $x \mapsto w'$, $y \mapsto v'$, *et* $z \mapsto u'$,
- *chaque arc e de* G_1 *est envoyé sur l'unique arc de* G_2 *dont les nœuds source et cible sont les images des nœuds source et cible de e.*

Remarquons que nous avons bien $m_1 = m_2 \circ m$ *et que les graphes sources des morphismes* m_1 *et* m_2 *sont effectivement bipartis au regard de la partition des nœuds induite par leurs images dans* $\{b, n\}$.

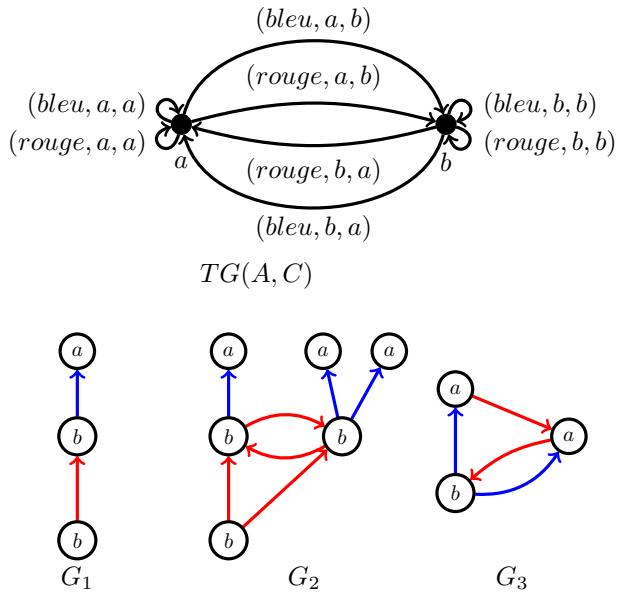


FIGURE 4.8 – *Graphes étiquetés comme des graphes typés*

Les principales propriétés catégorielles des graphes se transposent aux graphes typés : $(\mathrm{TG}, 1_{\mathrm{TG}})$ est terminal dans $\mathbf{Graph}_{\mathrm{TG}}$, $(G_\emptyset, G_\emptyset \to \mathrm{TG})$ est initial dans $\mathbf{Graph}_{\mathrm{TG}}$, les monomorphismes (resp. épimorphismes) des graphes typés sont les fonctions injectives (resp. surjectives) et l'existence des produits, produits fibrés, et sommes amalgamées est assurée.

Les graphes typés sont une généralisation des graphes étiquetés. En effet, étant donné un couple d'alphabet $\Sigma = (\Sigma_V, \Sigma_E)$, la catégorie Σ-**Graph** est équivalente à la catégorie $\mathbf{Graph}_{\mathbf{TG}(\Sigma)}$ La catégorie des graphes étiquetés sur Σ est équivalente à la catégorie $\mathbf{Graph}_{\mathbf{TG}(\Sigma)}$ des graphes typés sur le graphe $TG(\Sigma) = (V_\Sigma, E_\Sigma, s_\Sigma, t_\Sigma)$ défini par :

— $V_\Sigma = \Sigma_V$;
— $E_\Sigma = \Sigma_E \times \Sigma_V \times \Sigma_V$;
— $s_\Sigma \colon E_\Sigma \to V_\Sigma$; $(e, a, b) \mapsto a$;
— $t_\Sigma \colon E_\Sigma \to V_\Sigma$; $(e, a, b) \mapsto b$.

Exemple 4.4.12. *Reprenons les alphabets* $\mathbb{A} = \{a, b, \ldots, z\}$ *et* $\mathbb{C} = \{bleu, rouge, noir, vert, \ldots\}$. *Dans l'exemple 4.4.7, nous avions utilisé ces alphabets pour obtenir des graphes étiquetés par des lettres sur les nœuds et par des couleurs sur les arcs. Avec la construction décrite précédemment, on obtient alors le graphe type* $\mathrm{TG}(\mathbb{A}, \mathbb{C})$ *de la figure 4.8.*

On peut alors construire les graphes étiquetés G_1, G_2 *et* G_3 *comme des graphes typés par* $\mathrm{TG}(\mathbb{A}, \mathbb{C})$, *i.e., comme des morphismes vers* $\mathrm{TG}(\mathbb{A}, \mathbb{C})$.

Notons que cette description met aussi en lumière que les graphes typés sont essentiellement une caractérisation plus fines des graphes étiquetés, en lien avec la structure du graphe type considéré. En effet, enlever des arcs à un graphe $TG(\Sigma)$ permet d'interdire certaines relations de types. Par exemple, si nous enlevons les quatre boucles du graphe $\mathrm{TG}(\mathbb{A}, \mathbb{C})$, il n'est plus possible de typer le graphe G_3.

Pour conclure sur les graphes typés, nous tenons à rappeler, qu'un graphe typé est un morphisme vers un graphe type, que l'on représente souvent comme un graphe dont les éléments sont annotés par leur type. De plus, étant donné un graphe quelconque, il peut exister un, plusieurs ou aucun morphisme vers un graphe type qui décrivent autant de graphes typés.

4.4.5 Catégorie des graphes attribués

Les deux constructions précédentes (graphes étiquetés et typés) permettent d'attacher des informations aux éléments d'un graphe. Dans le cas des étiquettes, des méthodes de renommage existent, mais elles ne permettent pas d'encapsuler nativement des expressions de calcul sur les valeurs associées. Dans le cas des types, ceux-ci sont par nature immuables. L'introduction de décorations appelées *attributs* est une réponse à cette préoccupation. Avant d'introduire les graphes attribués, nous rappelons quelques notions autour des types de données.

Les types de données décrivent les structures de données en termes de syntaxe et de sémantique. La syntaxe est déterminée par une signature décrivant les types de données et les opérations disponibles sur les données. La sémantique est décrite par une algèbre qui fournit des ensembles de valeurs pour chacun des types et des interprétations fonctionnelles des opérations. L'algèbre des termes joue un rôle particulier, en faisant un pont entre syntaxe et sémantique par l'ajout de variables liées aux types permettant d'obtenir des termes.

Définition 4.4.13 (Signature)**.** *Une signature $\Omega = (S, F)$ d'un type de données est définie par un ensemble S de noms de types et un ensemble F de noms d'opérations munis d'un profil dans $S^* \times S$. Un nom d'opération f munie d'un profil $(s_1, \ldots, s_m, s)$ est noté $f : s_1 \times \ldots \times s_m \to s$ et est dit d'arité m.*

Définition 4.4.14 (Ω-algèbre)**.** *Étant donnée une signature Ω, une Ω-algèbre est définie par une famille d'ensembles de valeurs $(\mathcal{A}_s)_{s\in S}$ et par une fonction $f^{\mathcal{A}} : \mathcal{A}_{s_1} \times \ldots \times \mathcal{A}_{s_m} \to \mathcal{A}_s$ pour tout nom d'opération $f : s_1 \times \ldots \times s_m \to s$ de Ω.*

Un morphisme de Ω-algèbres $g : \mathcal{A} \to \mathcal{B}$ est une famille d'applications $(g_s : \mathcal{A}_s \to \mathcal{B}_s)_{s\in S}$ vérifiant que pour tout $f : s_1 \times \ldots \times s_m \to s \in F$ et pour $a_1 \in \mathcal{A}_{s_1}$, …, $a_m \in \mathcal{A}_{s_m}$, $g_s(f^{\mathcal{A}}(a_1, \ldots, a_m)) = f^{\mathcal{B}}(g_{s_1}(a_1), \ldots, g_{s_m}(a_m))$.

Les Ω-algèbres et leurs morphismes forment une catégorie $\mathbf{Alg}(\Omega)$.

Définition 4.4.15 (Ω-termes). *Soit* $\Omega = (S, F)$ *une signature et* X *une famille* S*-indexée d'ensembles de variables, disjointe de* F*. Pour* $s \in S$*, l'ensemble* $T_\Omega(X)_s$ *des termes de type* s *sur* Ω *avec variables dans* X *est défini inductivement par :*

— *pour* x *dans* X_s*,* x *est un terme de* $T_\Omega(X)_s$*;*

— *pour* $f : s_1 \times \ldots \times s_m \to s$ *dans* F*, pour tout* $t_1 \in T_\Omega(X)_{s_1}$*, …,* $t_m \in T_\Omega(X)_{s_m}$*,* $f(t_1, \ldots, t_m)$ *est un terme de* $T_\Omega(X)_s$*.*

On note $t : s$ *pour* $t \in T_\Omega(X)_s$ *et* $\mathrm{Var}(t)$ *pour l'ensemble des variables de* t*. L'ensemble* $T_\Omega(X) = \bigsqcup_{s \in S} T_\Omega(X)_s$ *est l'ensemble des* Ω*-termes avec variables dans* X*.*

Les termes définissent une algèbre, dite *algèbre des termes* et notée $\mathcal{T}_\Omega(X)$ où l'ensemble de valeurs $\mathcal{T}_\Omega(X)_s$ de type s est l'ensemble des termes lui-même $T_\Omega(X)_s$ et où, pour toute opération $f : s_1 \times \ldots \times s_m \to s$ dans F, pour tous $t_1 \in T_\Omega(X)_{s_1}$, …, $t_m \in T_\Omega(X)_{s_m}$, $f^{\mathcal{T}_\Omega(X)}(t_1, \ldots, t_m)$ est le terme $f(t_1, \ldots, t_m)$ de $T_\Omega(X)_s$. Pour une Ω-algèbre $\mathcal{A}$, toute fonction $f : X \to \mathcal{A}$ définit un morphisme d'algèbres unique $f^* : \mathcal{T}_\Omega(X) \to \mathcal{A}$ exprimant l'évaluation des termes dans $\mathcal{A}$.

Par ailleurs, la catégorie $\mathbf{Alg}(\Omega)$ a pour objet terminal $\mathbb{1}_{\mathbf{Alg}(\Omega)}$ ayant pour ensembles de valeurs un singleton pour chaque type s et pour toute opération $f : s_1 \times \ldots \times s_m \to s$, l'unique fonction d'interprétation définie sur ces valeurs.

Exemple 4.4.16. *A des fins d'illustration, nous considérerons dans ce chapitre la signature* $\Omega = (\{bool, int\}, \{true, false :\to bool, 0, 1, \ldots :\to int, + : int \times int \to int\})$ *contenant les types de données des booléens et des entiers, munis des constantes usuelles et de l'addition de deux entiers. Nous considérerons l'algèbre* $\mathcal{A}$ *définie par* $\mathcal{A}_{bool} = \{true, false\}$ *et* $\mathcal{A}_{int} = \mathbb{N}$ *et l'interprétation de l'opération* $+$ *par l'addition dans* $\mathbb{N}$*.*

Les algèbres permettent d'exprimer des calculs sur des données. Il reste à savoir comment les graphes sont annotés avec ce type d'informations. Par simplicité, nous définissons les graphes attribués pour les seuls nœuds, sans attribut pour les arcs. La première étape consiste à donner un support pour décrire l'ajout des attributs. Ce support, appelé E-graphe, étend la construction d'un graphe avec un ensemble de nœuds de données D et un ensemble d'arcs d'attribution A.

$$E \underset{t}{\overset{s}{\rightrightarrows}} V \xleftarrow{sa} A \xrightarrow{ta} D \tag{4.4}$$

Définition 4.4.17. *Un E-graphe* $G = (V_G, D_G, E_G, A_G, s_G, t_G, sa_G, ta_G)$ *est défini par un graphe* (V_G, E_G, s_G, t_G)*, un ensemble de données* D_G*, un ensemble d'arcs d'attribution* A_G *et deux fonctions* $sa_G \colon A_G \to V_G$ *et* $ta_G \colon A_G \to D_G$ *pour les nœuds sources et cibles des arcs d'attribution.*

Un morphisme d'E-graphes $m \colon G \to H$ *est défini par 4 fonctions* $m_V \colon V_G \to V_H$, $m_D \colon D_G \to D_H$, $m_E \colon E_G \to E_H$, *et* $m_A \colon A_G \to A_H$ *qui induisent des diagrammes commutatifs dans* **Set** *pour les fonctions sources et cibles des arcs et des arcs d'attribution. Les E-graphes et leurs morphismes forment une catégorie notée* **EGraph**.

Les diagrammes commutatifs induits par un morphisme de E-graphes sont analogues à ceux d'un morphisme de graphes (voir diagrammes 4.3). On obtient la construction des graphes attribués à partir des E-graphes par l'ajout d'une Ω-algèbre dont l'ensemble des valeurs coïncide avec l'ensemble de données D. Dans la suite, on suppose donnée une signature $\Omega = (S, F)$ et une Ω-algèbre $\mathcal{A}$.

Définition 4.4.18 (Graphes attribués)**.** *Un graphe* Ω*-attribué* $AG = (G, \mathcal{A})$ *est défini par un E-graphe* $G = (V_G, D_G, E_G, A_G, s_G, t_G, sa_G, ta_G)$ *et une* Ω*-algèbre* $\mathcal{A}$ *vérifiant* $D_G = \bigsqcup_{s \in S} \mathcal{A}_s$.

Un morphisme de graphes Ω*-attribués* $m \colon (G, \mathcal{A}) \to (H, \mathcal{B})$ *est un couple* $(m_G, m_{\mathcal{A}})$ *avec* $m_G \colon G \to H$ *un morphisme de E-graphes et* $m_{\mathcal{A}} \colon \mathcal{A} \to \mathcal{B}$ *un morphisme d'algèbres assurant que le diagramme ci-dessous commute pour tous les types* s *de* S*,*

$$\begin{array}{ccc} \mathcal{A}_s & \xrightarrow{(m_{\mathcal{A}})_s} & \mathcal{B}_s \\ \downarrow & & \downarrow \\ D_G & \xrightarrow[(m_G)_D]{} & D_H \end{array}$$

où les flèches verticales représentent des inclusions dans **Set**.

Les graphes Ω*-attribués et leurs morphismes forment la catégorie* Ω-**AGraph**.

Il est à noter que l'ensemble D de données correspond à l'union des ensembles de valeurs de l'algèbre utilisée pour attribuer le graphe. Cet ensemble est souvent infini, on se contentera alors de représenter les éléments de D cibles d'un arc d'attribution.

Exemple 4.4.19. *La figure 4.9 illustre la notion de graphe attribué défini à l'aide de l'*Ω*-algèbre* $\mathcal{A}$ *donnée dans l'exemple 4.4.16. L'ensemble de données est* $D = \{true, false\} \sqcup \mathbb{N}$. *Le graphe sous-jacent est dessiné en noir, les arcs d'attri-*

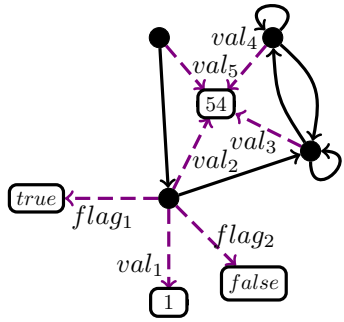


FIGURE 4.9 – *Un graphe attribué*

bution sont dessinés en violet pointillé et les données sont à l'intérieur des nœuds elliptiques. Ici l'ensemble des arcs d'attribution est $\{val_i \mid i \in 1..5\} \cup \{flag_j \mid j \in 1..2\}$ *où les arcs* val_i *ont pour cible un entier et les arcs* $flag_j$ *un booléen.*

Dans les graphes attribués, n'importe quelle donnée peut être associée à n'importe quel nœud. Comme précédemment, l'utilisation d'un graphe type permet de catégoriser les attributs selon les types de données. Notons qu'un graphe type pour la catégorie des graphes attribués comporte nécessairement un E-graphe et une Ω-algèbre. Le rôle de l'algèbre n'est ici que de caractériser les types de données portées par un nœud, autrement dit l'algèbre associée est nécessairement l'algèbre finale $\mathbb{1}_{\mathbf{Alg}(\Omega)}$.

Définition 4.4.20 (Graphes attribués typés)**.** *Un graphe type* Ω*-attribué est un graphe* Ω*-attribué* $\mathrm{ATG} = (\mathrm{TG}, \mathbb{1}_{\mathbf{Alg}(\Omega)})$.

La catégorie Ω-$\mathbf{AGraph}_{\mathbf{ATG}}$ *des graphes attribués typés sur le graphe attribué type* ATG *est la catégorie slice* Ω-$\mathbf{AGraph}/\mathrm{ATG}$.

Chaque arc d'un graphe type Ω-attribué $\mathrm{ATG} = (\mathrm{TG}, \mathbb{1}_{\mathbf{Alg}(\Omega)})$ permet de décrire les attributs associés aux nœuds par un identificateur, l'arc lui-même, et par un type de donnée, la valeur dans $\mathbb{1}_{\mathbf{Alg}(\Omega)}$ cible de l'arc. La figure 4.10 contient trois exemples de graphes Ω-attribués types. Les exemples concernent différentes signatures de données. Rappelons que pour une signature Ω, l'algèbre finale $\mathbb{1}_{\mathbf{Alg}(\Omega)}$ a pour ensembles de valeurs un singleton pour chacun des types s de Ω que l'on note $\{s\}$ par convention. Les éléments des nœuds elliptiques des graphes de la figure 4.10 coïncident ainsi avec des noms de types de la signature considérée Ω.

Exemple 4.4.21. *ATG_1 est un graphe* Ω*-attribué défini pour la signature* $\Omega = \{bool, int\}$ *introduite dans l'exemple 4.4.16. Nous définissons un morphisme de graphes* Ω*-attribués du graphe de la figure 4.9 vers ATG_1 comme attendu : les*

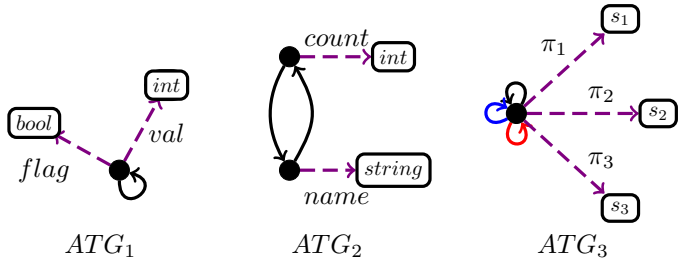


FIGURE 4.10 – *Exemples de graphes types pour définir des graphes attribués typés*

*nœuds et arcs noirs sont envoyés sur le nœud et la boucle noirs de ATG_1, les arcs pointillés violets val_i pour $i \in \{1,2,3,4,5\}$ (resp. $flag_j$ pour $j \in \{1,2\}$) sont envoyés sur l'arc pointillé violet val (resp. $flag$) partageant le type des nœuds cibles. Ce morphisme définit un élément de Ω-**AGraph**$/ATG_1$. Soulignons qu'il est d'usage de nommer les arcs des graphes attribués types à l'aide d'identifiants significatifs au regard du contexte. Ainsi, l'arc « val » associe une valeur entière aux nœuds du graphe tandis que l'arc « $flag$ » leur associe une valeur booléenne.*

Exemple 4.4.22. *ATG_2 est un graphe Ω_2-attribué défini pour la signature $\Omega_2 = \{string, int\}$ où $string$ décrit le type de données des chaînes de caractères. Ce graphe type peut être utilisé pour caractériser la classe des réseaux de Petri : le graphe sous-jacent décrit les graphes bipartis comme dans la figure 4.6 séparant les nœuds places et les nœuds transitions. Il y a un type d'arcs d'attribution par type de nœud : les nœuds places ont des attributs « count » de type « int » utiles pour indiquer le nombre de jetons présents et les nœuds transitions ont des attributs « name » de type « string » pour nommer l'action effectuée lorsque la transition est tirée.*

Exemple 4.4.23. *ATG_3 est un graphe Ω_3-attribué défini pour la signature $\Omega_3 = \{s_1, s_2, s_3\}$ et pour des graphes dont les arcs sont typés par les couleurs $\{rouge, noir, bleu\}$. ATG_3 préfigure les graphes Ω-attribués utilisés pour caractériser la classe des cartes combinatoires. En utilisant une couleur par dimension, les arcs colorés permettent d'indiquer quels sont les voisins d'un nœud dimension par dimension. Selon le contexte, les arcs d'attribution π_1, π_2 et π_3 permettent de décrire différents plongements géométriques : couleur, position, orientation, normale, concentration, forces... Les types de données s_1, s_2 et s_3 sont alors spécifiés au cas par cas.*

Remarques :

— Chaque arc d'attribution d'un graphe Ω-attribué type décrit un attribut par un nom, le nom de l'arc lui-même, et par un ensemble de valeurs, toutes les valeurs de type la cible de l'arc d'attribution.

Ainsi deux arcs d'attribution avec même source et même cible dans un graphe type indiquent l'emploi de deux attributs de même profil.

— Intuitivement, les graphes typés ont une structure conforme à celle du graphe type au sens de l'inclusion. Tous les arcs et arcs d'attribution des graphes typés sont présents dans le graphe type. Néanmoins, rien ne contraint qu'il existe systématiquement dans les graphes typés une contrepartie à tous les éléments du graphe type. Par exemple, à partir d'un nœud particulier, il peut ne pas exister d'arc coloré ou d'arc d'attribution. De même, il peut exister plusieurs arcs colorés ou arcs d'attribution attachés à un même nœud. C'est précisément le cas du graphe attribué de la figure 4.9 typé par le graphe attribué type ATG_1 de la figure 4.10.

4.5 Transformations de graphes

De même que la réécriture de termes permet de transformer localement un terme, e.g. $(x + 0) \times y$, en identifiant une expression d'intérêt, e.g. l'expression $(x + 0)$, pour la remplacer par une autre expression, e.g. x, pour produire un nouveau terme, dans le cas présent, le terme $x \times y$, les principales approches de transformations de graphes ont pour objectif général de réécrire les graphes en remplaçant un motif d'intérêt L par un nouveau motif R. Tandis que la réécriture de termes se fait sur place, à la position du sous-terme identifié comme instance de l'expression d'intérêt, la réécriture de graphes nécessite plus de soin pour dans un premier temps, séparer le motif d'intérêt du graphe d'origine et dans un second temps, raccrocher le nouveau motif au graphe amputé du motif d'intérêt. Les déconnexions et reconnexions de morceaux de graphes seront exprimées à l'aide des morphismes de graphes et des constructions catégorielles présentées dans la section 4.3.

Parmi les différentes approches de transformation de graphes, nous présentons celle connue sous le nom « double somme amalgamée »[7]. Cette approche exploite l'identification d'un sous-graphe K, appelé interface, qui explicite l'accroche du motif d'intérêt L et du nouveau motif R au graphe sous transformation G. Nous en présentons une version restrictive où tous les morphismes sont des monomorphismes. L'utilisation de monomorphismes, approche courante dans les catégories adhésives présentées dans la suite, interdit le repliement des motifs manipulés. On assure ainsi la réversibilité des transformations tout en étant un prérequis à certaines

7. Approche communément dénotée par le sigle DPO pour Double PushOut.

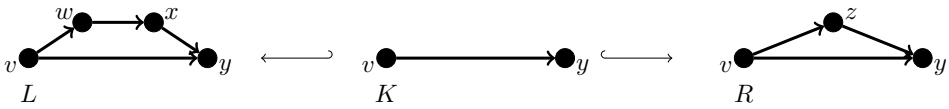


FIGURE 4.11 – *Une règle de réécriture de graphe*

constructions plus élaborées et tout en garantissant le déterminisme de la réécriture. De manière très pragmatique, il s'agit aussi de l'approche la plus simple pour le raisonnement lors de la construction du graphe transformé puisqu'elle supporte une vision à l'échelle des éléments du graphe.

4.5.1 Réécriture dans Graph

Nous commençons par étudier la réécriture par double sommes amalgamées dans la catégorie **Graph** qui admet une description entièrement constructive de l'étape de réécriture.

Définition 4.5.1 (Règle)**.** *Une règle, ou production, p est un span $L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R$ de monomorphismes.*

Exemple 4.5.2. *La règle de la figure 4.11 permet de transformer un chemin de longueur* 3 *parallèle à un arc en un chemin de longueur* 2*. Notons que la règle supprime les deux nœuds du chemin pour en créer un nouveau, ce qui impose que les deux nœuds (w et x) puissent être supprimés sans détruire la structure plus générale du graphe.*

Les graphes L, K et R sont respectivement appelés *membre gauche*, *interface* et *membre droit* de la règle, et décrivent respectivement le sous-graphe modifié par p, le sous-graphe préservé lors de l'application de p et le graphe ajouté dans le graphe transformé. Intuitivement, l'application d'une règle par DPO à un graphe G peut être résumée en trois étapes :

1. une occurence de L est recherchée dans G ;
2. les éléments (nœuds et arcs) de L qui ne sont pas dans K sont supprimés ;
3. les éléments de R qui ne sont pas dans K sont ajoutés.

Plus formellement, l'occurence de L dans G est définie par le biais d'un monomorphisme $m\colon L \hookrightarrow G$ appelé *match* de l'application. La suppression des éléments de L non présents dans K est obtenue par la construction d'un complément de somme amalgamée, noté D, à partir des graphes K, L et G et des morphismes les reliant $l\colon K \hookrightarrow L$ et $m\colon L \hookrightarrow G$, et donnant

lieu à la construction d'une paire de morphismes $k\colon K \to D$ et $g\colon D \to G$. Enfin, l'ajout des éléments de R est obtenu par construction d'une somme amalgamée à partir du span $D \overset{k}{\leftarrow} K \overset{r}{\rightarrow} R$. Cependant, en toute généralité, l'existence d'un complément de somme amalgamée D et des morphismes associés n'est pas assurée. Elle est assujettie à la *condition de recollement* donnée ci-dessous :

Définition 4.5.3 (Match)**.** *Un match pour une règle $p = L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R$ est un monomorphisme $m\colon L \hookrightarrow G$. Un match satisfait la* condition de recollement *si $K \overset{l}{\hookrightarrow} L \overset{m}{\hookrightarrow} G$ admet un complément de somme amalgamée, c'est-à-dire s'il existe un objet D et une paire de morphismes $K \overset{k}{\rightarrow} D \overset{g}{\rightarrow} G$ tel que le diagramme suivant est une somme amalgamée :*

$$\begin{array}{ccc} L & \overset{l}{\hookleftarrow} & K \\ {\scriptstyle m}\downarrow & PO & \downarrow{\scriptstyle k} \\ G & \underset{g}{\leftarrow} & D \end{array}$$

Dans **Graph**, la construction des sommes amalgamées par union quotientée de **Set** (cf Exemple 4.3.9) se généralise en raisonnant composante par composante, c'est-à-dire en prenant la somme amalgamée de l'ensemble des sommets et celle de l'ensemble des arcs. L'injectivité induite par l'utilisation de monomorphismes permet alors d'établir une condition pratique équivalente à la condition de recollement appelée condition d'arcs pendants. Aucun nœud de $m(V_L) \setminus m(l(V_K))$ est la source ou la cible d'un arc de $E_G \setminus m(E_L)$.

A partir de l'occurrence d'un match du membre gauche de la règle dans un graphe G, la transformation de G est décrite par une étape de dérivation directe.

Définition 4.5.4 (Dérivation directe)**.** *Une règle $p = L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R$ transforme un graphe G en un graphe H via un match $m\colon L \hookrightarrow G$ s'il existe un diagramme*

$$\begin{array}{ccccc} L & \overset{l}{\hookleftarrow} & K & \overset{r}{\hookrightarrow} & R \\ {\scriptstyle m}\downarrow & PO & \downarrow & PO & \downarrow{\scriptstyle m'} \\ G & \leftarrow & D & \rightarrow & H \end{array}$$

où les deux carrés sont des sommes amalgamées. De plus, la dérivation directe $G \Rightarrow^{r,m} H$ existe et est unique (à isomorphisme près) si m satisfait la condition de recollement. Le morphisme $m'\colon R \hookrightarrow H$ est appelé comatch de la dérivation.

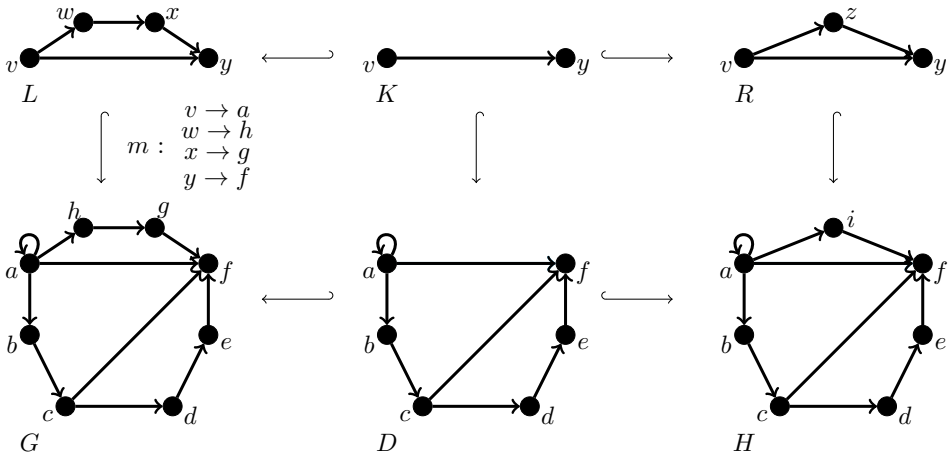


FIGURE 4.12 – *Une dérivation directe*

Exemple 4.5.5. *La figure 4.12 illustre une dérivation directe de la règle de la figure 4.11. Cette dérivation est construite à partir du match induit par la fonction* $v \mapsto a$, $w \mapsto h$, $x \mapsto g$, $y \mapsto f$ *sur les nœuds. Ici* $K \overset{l}{\hookrightarrow} L \overset{m}{\hookrightarrow} G$ *admet un complément de somme amalgamée, ce que l'on peut vérifier en observant que l'image des nœuds supprimés w et x, c'est-à-dire les nœuds h et g, apparaissent dans l'occurence de L avec tous leurs arcs incidents. On peut alors construire le complément de somme amalgamée* $K \hookrightarrow D \hookrightarrow G$ *en supprimant h et g et leurs arcs incidents de G pour obtenir D et les morphismes évidents de K vers D d'une part et de D vers G d'autre part. La deuxième partie de la dérivation est construite de manière analogue en ajoutant les éléments créés de R, i.e., z ainsi que les arcs de v à z et de z à y. On rappelle que cette seconde partie existe toujours, puisque tout span admet une somme amalgamée dans* **Graph**.

Exemple 4.5.6. *La figure 4.13 illustre une deuxième dérivation directe de la règle de la figure 4.11 sur le même graphe G. Ici le match est induit par la fonction* $v \mapsto c$, $w \mapsto d$, $x \mapsto e$, $y \mapsto f$ *sur les nœuds. Par similitude, on construit le graphe D par délétion des éléments supprimés de L auquel on ajoute ensuite les éléments créés de R pour obtenir H, le résultat de la dérivation directe.*

4.5.2 Catégories adhésives

La réécriture par double somme amalgamée est bien fondée dans la catégorie **Graph**. Cependant, le complément de la somme amalgamée n'est pas caractérisé par une construction universelle, il peut donc ne pas être unique dans une catégorie quelconque. Dans ce cas, l'application d'une

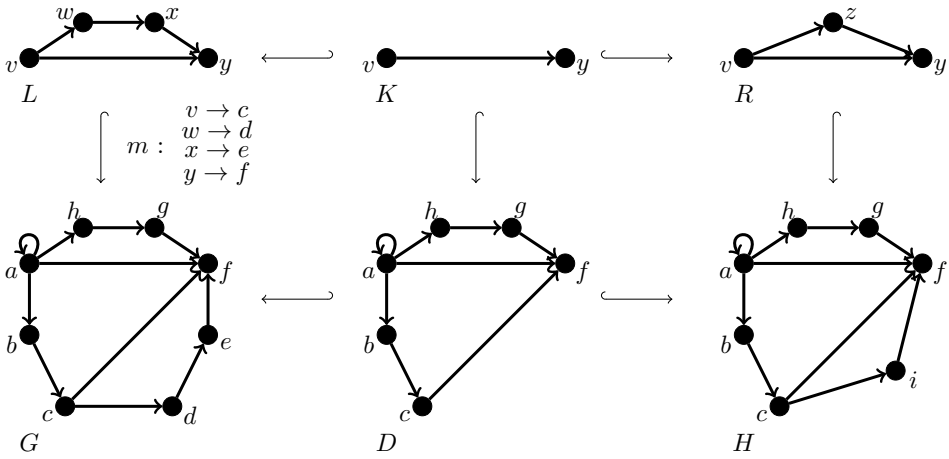


FIGURE 4.13 – *Une deuxième dérivation directe construite à partir de la même règle*

règle devient non-déterministe. Plusieurs tentatives d'axiomatisation ont été proposées pour remédier à ce problème. Nous présentons ici celle des catégories adhésives. De plus, il existe différentes variations autour de cette notion d'adhésivité mais l'idée générale demeure que les sommes amalgamées (et les produits fibrés) se comportent comme dans **Set**, assurant le bien-fondé de la réécriture. Les implications liées à ces catégories dépassent le cadre de ce chapitre en permettant notamment d'obtenir des propriétés de confluence pour les grammaires obtenues.

L'adhésivité d'une catégorie repose sur l'existence de produits fibrés, de sommes amalgamées le long des monomorphismes et d'une caractérisation des sommes amalgamées le long des monomorphismes comme carrés de van Kampen dont la construction complète ne sera pas abordée dans ce chapitre.

L'adhésivité permet d'obtenir un ensemble de lemmes :

— stabilité des monomorphismes par somme amalgamées : la somme amalgamée du span $A \hookleftarrow C \rightarrow B$ (où $C \hookrightarrow A$ est un monomorphisme) est un cospan $A \rightarrow D \hookleftarrow B$ (où $B \hookrightarrow D$ est un monomorphisme).

$$\begin{array}{ccc} C & \hookrightarrow & A \\ \downarrow & & \downarrow \\ B & \hookrightarrow & D \end{array}$$

Ce premier lemme assure qu'un match injectif donne un comatch injectif, éliminant de fait la possibilité de fusionner des éléments du motif droit de la règle dans le graphe transformé.

— unicité du complément de somme amalgamée (à isomorphisme près). Ce second lemme assure le déterminisme de la réécriture : étant donné une règle et un match, il n'existe qu'un résultat de la dérivation (à isomorphisme près).

De plus, l'adhésivité est préservé par slicing, ce qui assure que les transformations de graphes typés (et donc étiquetés) par DPO sont bien définies.

Cependant, certaines catégories intéressantes ne sont pas adhésives. C'est par exemple le cas des graphes attribués (ou de leur extension avec un graphe type attribué). Dans de tels cas, il est possible de réduire la famille de monomorphismes à considérer. On considère alors un système stable de monomorphismes.

Définition 4.5.7 (Système stable de monomorphismes)**.** *$\mathscr{M}$ est un système stable de monomorphismes si $\mathscr{M}$ est un système de monomorphismes contenant l'ensemble des isomorphismes, stable par composition, somme amalgamée et produit fibré.*

Dans le cas des graphes attribué, le système stable de monomorphismes correspond aux monomorphismes pour lesquels le morphisme d'algèbres est un isomorphisme. De tels morphismes permettent de considérer une algèbre immuable par réécriture, ce qui permet de changer les valeurs sans modifier les règles de calcul dans l'algèbre.

Propriété 4.5.8 (Classe $\mathscr{M}$ pour les graphes attribués)**.** *La classe $\mathscr{M}$ contenant les morphismes $m = (m_G, m_\mathcal{A})$ de Ω-**AGraph** où m_G est un monomorphisme et $m_\mathcal{A}$ est un isomorphisme constitue un système stable de monomorphismes.*

Une règle de graphes attribués est alors un span de $\mathscr{M}$-morphismes entre des graphes attribués sur $\mathcal{T}_\Omega(X)$, l'Ω-algèbre de termes avec variables dans X.

Définition 4.5.9 (Règle de graphes attribués)**.** *Une règle Ω-attribuée est un span $L \hookleftarrow K \hookrightarrow R$ de $\mathscr{M}$-morphismes dans Ω-**AGraph** où les graphes attribués partagent l'algèbre $\mathcal{T}_\Omega(X)$ et les morphismes d'algèbres sont $1_{\mathcal{T}_\Omega(X)}$.*

La construction s'étend aux graphes typés attribués par slicing sur le graphe type attribué $\mathrm{ATG} = (\mathrm{TG}, \mathbf{1}_{\mathbf{Alg}(\Omega)})$.

Exemple 4.5.10. *Pour interpréter la règle de la figure 4.14, il convient de donner la signature sous-jacente. Il est ici possible de considérer $\Omega_{int} = (\{int\}, \{+ : int \times int \to int\})$ et l'ensemble $\mathbb{A}$ des lettres de l'alphabet comme ensemble de*

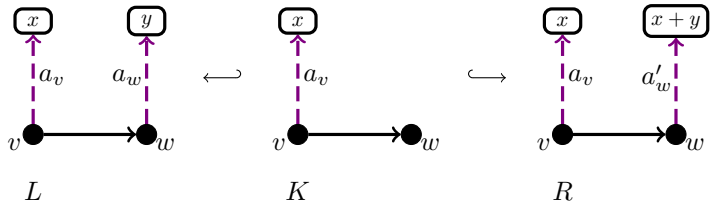


FIGURE 4.14 – *Une règle de graphes attribués*

variables, toutes de type int. Une catégorie de graphe Ω_{int}-attribués peut alors être construite en considérant l'algèbre $\mathcal{A}_{\mathbb{N}}$ qui possède $\mathbb{N}$ comme ensemble de valeurs pour $\mathcal{A}_{int}$ et interprète $+$ comme l'addition dans $\mathbb{N}$. Cependant, la description d'une règle et donc le sens qu'il faut lui donner n'est pas sémantique mais bien syntaxique puisque les graphes de la règle sont construits sur l'algèbre des termes $\mathcal{T}_{\Omega_{int}}(\mathbb{A})$. Intuitivement, la règle change l'attribut associé à w par la somme avec l'attribut associé à v. Concrètement, les termes y et $x + y$ existant tous les deux dans $\mathcal{T}_{\Omega_{int}}(\mathbb{A})$, et donc dans L, K et R. Ainsi, la règle ne modifie pas de nœud mais correspond à la suppression de l'arc d'attribution de source w de L et à la création d'un nouvel arc d'attribution de même source dans R.

La réécriture par double somme amalgamée et la construction d'une dérivation directe s'étend alors aux graphes (typés) attribués. On récupère aussi les lemmes précédents. Notons que dans la dérivation directe $G \Rightarrow^{r,m} H$ de graphes attribués, le match $m\colon L \hookrightarrow G$ fournit l'évaluation des termes de L par l'association des variables de X à des valeurs concrètes dans l'algèbre de G.

Exemple 4.5.11. *Nous donnons à la figure 4.15 un exemple d'application de la règle donnée en figure 4.14. Les morphismes $L \hookrightarrow G$ (match), $K \hookrightarrow D$, et $R \hookrightarrow H$ (comatch) sont définis en envoyant le nœud v sur a et le nœud w sur b dans les différents graphes G, D et R. Les images des arcs de L, K et R se déduisent de ces associations : par exemple, l'arc $v \to w$ est envoyé sur l'arc $b \to a$. En remarquant que l'arc d'attribution a_v dans L est envoyé sur l'arc d'attribution a_b, le morphisme d'algèbres associé associe la valeur 8 à la variable x. De même, via l'arc d'attribution a_w, on en déduit que la valeur 4 est associé à la variable y. Ainsi, l'évaluation du terme $x + y$ par le morphisme fournit la valeur 12. Finalement, l'arc d'attribution de source a et de cible 4 dans G est retiré, et un nouvel arc d'attribution de source a est ajouté dans R avec pour cible la valeur 12.*

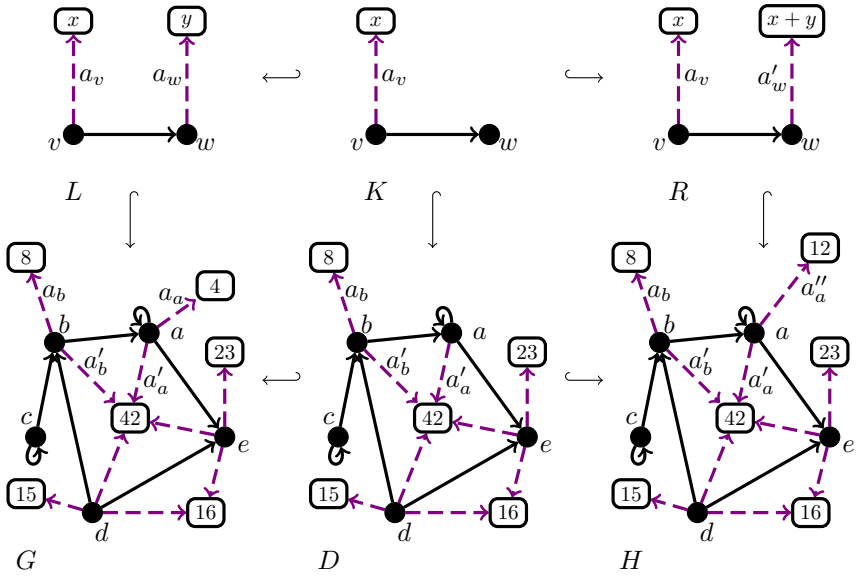


FIGURE 4.15 – *Une réécriture de graphes attribués*

4.6 Application à la modélisation géométrique

Dans cette section, nous discutons d'applications de la théorie de réécriture de graphes décorés pour la modélisation géométrique, tout d'abord par la vision d'un graphe comme un maillage [8], puis via le modèle des cartes généralisées interprétées comme des graphes.

4.6.1 Maillages

Un maillage surfacique est décrit par un ensemble de sommets, d'arêtes et de faces. En oubliant dans un premier temps les faces, que l'on peut par exemple décrire comme des cycles d'arêtes, on peut représenter un maillage comme un graphe non-orienté. À cette information topologique, on ajoute généralement différentes informations, typiquement portées par les sommets du maillage : position géométrique, normale, couleur, position dans une texture… Pour encoder ces informations, il y a essentiellement deux choix : l'utilisation d'étiquettes ou d'attributs. Dans l'optique de pouvoir modifier facilement les maillages, il est préférable de représenter ces informations comme des types de données autorisant des calculs, autrement dit via des attributs.

8. Il faudrait en réalité parler de graphe plongé sur une surface, ce qui nous rapproche d'ailleurs de la notion de carte évoquée par la suite.

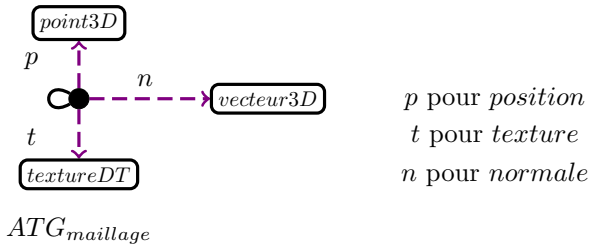


FIGURE 4.16 – *Graphe attribué type pour encoder les maillages.*

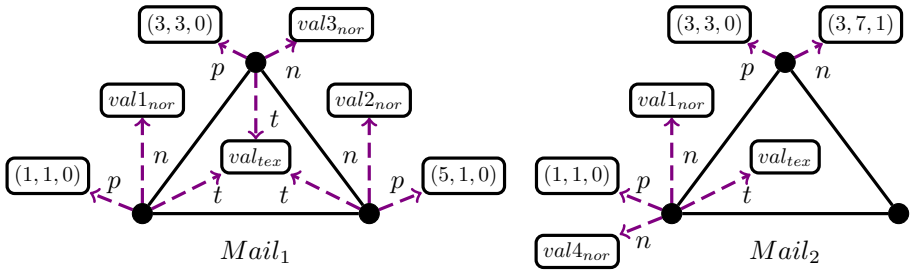


FIGURE 4.17 – *Deux exemples de graphes attribués typés par* $ATG_{maillage}$

Afin de pouvoir manipuler les attributs en fonction de la signification que l'on veut leur donner, plutôt que par la simple description du type de données associé, on considère la classe des graphes attribués typés par $ATG_{maillage}$ donné en figure 4.16. Le graphe attribué type $ATG_{maillage}$ indique qu'un nœud du maillage peut être relié à n'importe quel autre nœud du maillage et qu'il peut être décoré avec trois attributs, identifiés respectivement par les identificateurs *position*, *normale* et *texture* (dénotés respectivement par les lettres p, n et t pour des soucis de place) dont les types respectifs sont *point*3*D*, *vecteur*3*D* et *textureDT*. Les identificateurs sont choisis pour suggérer un type de données raisonnable pour décrire les maillages : pour être complet, il conviendrait néanmoins de spécifier complètement la signature et l'algèbre de calcul.

Nous donnons en figure 4.17 deux exemples de graphes attribués typés par $ATG_{maillage}$. Les valeurs d'attribution restent abstraites (e.g. $val1_{nor}$, val_{text}) et ont été choisies de façon complètement arbitraire. Notons que le maillage $Mail_1$ vérifie la propriété particulière d'*unicité des attributs* : tout nœud possède exactement un arc d'attribution par arc d'attribution du graphe attribué type $ATG_{maillage}$. En revanche, le maillage $Mail_2$ ne respecte aucune régularité : les nœuds possèdent 0, 1 ou plusieurs arcs d'at-

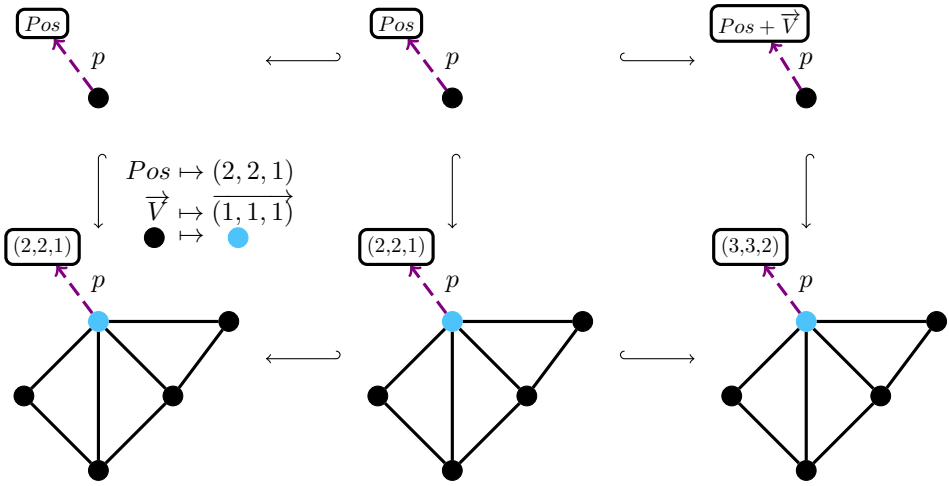


FIGURE 4.18 – *Déplacement d'un point du maillage*

tribution pour chaque arc d'attribution de $ATG_{maillage}$. Bien évidemment, les maillages bien formés sont les graphes attribués typés par $ATG_{maillage}$ vérifiant la propriété d'unicité des attributs.

La figure 4.18 décrit l'application d'une règle de déplacement d'un point du maillage. Les membres gauche et droit ainsi que l'interface contiennent un unique point avec l'arc d'attribution *position* explicitée : dans l'interface et membre gauche, l'attribut est une variable *Pos* de type *point3D* et dans le membre droit, l'attribut est le terme $Pos + \overrightarrow{V}$ avec $\overrightarrow{V}$ variable de type *vecteur3D*. Le morphisme d'algèbres du match est défini par les associations $P \mapsto (2,2,1)$, $\overrightarrow{V} \mapsto \overrightarrow{(1,1,1)}$ tandis que le morphisme de graphes associe le nœud bleu du graphe au nœud noir du membre gauche de la règle.

Alors que la règle de la figure 4.18 modifie les valeurs d'attribution d'un maillage sans en modifier la structure, la figure 4.19 décrit l'application d'une règle modifiant localement un maillage. Cette opération, appelée « edge flip », peut être utile pour rééquilibrer les triangles dont la géométrie serait jugée trop plate. Les nœuds et arcs bleus définissent les morphismes définissant l'application de la règle, la correspondance nœud à nœud étant donnée par leur position respective. Remarquons qu'aucun attribut n'apparaissant pas dans la règle, les nœuds préservés par la règle gardent leurs attributs inchangés. Dans le cas présent, par souci d'avoir des figures allégées, aucun attribut n'est mentionné sur le graphe.

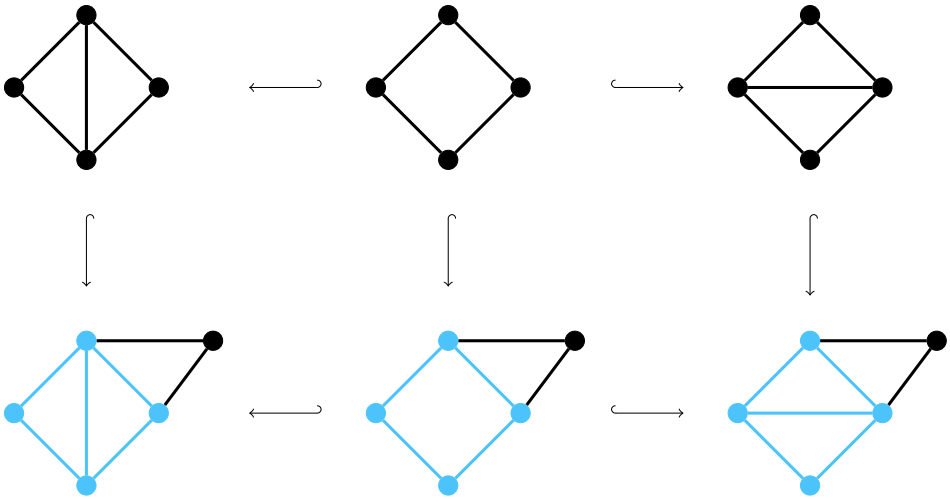


FIGURE 4.19 – *Opération « edge flip »*

Enfin, les deux règles des figures 4.18 et 4.19 préservent la propriété d'unicité des attributs, i.e. sont telles que l'application à des maillages vérifiant l'unicité des attributs produit des maillages vérifiant aussi la propriété des attributs. Autrement dit, par construction, ce sont des opérations internes dans la classe des maillages bien formés. En particulier, la règle de la figure 4.18 filtre l'attribut *position* pour le modifier.

4.6.2 Cartes généralisées plongées

Cette première vision de la description d'un objet géométrique reste relativement restrictive. En effet, la construction d'un maillage présuppose généralement une information supplémentaire permettant de reconstruire les faces. Cette information peut être explicite, e.g., liste des sommets de chaque face, ou implicite, e.g., ordre des arêtes autour d'un sommet. Dans le domaine de la modélisation géométrique à base topologique, les objets sont décrits par le biais d'une subdivision en cellules topologiques (sommets, arêtes, faces, volumes…). Dans une vision combinatoire, ces cellules sont ensuite reliées par le biais de relations d'incidence ou d'adjacence. Par exemple, les cartes généralisées (ou G-cartes) considèrent des brins décrivant des n-uplets de cellules topologiques et décrivent les relations d'incidences par le biais de permutations sur ces brins. La définition combinatoire des cartes généralisées est la suivante (cf. [6]) :

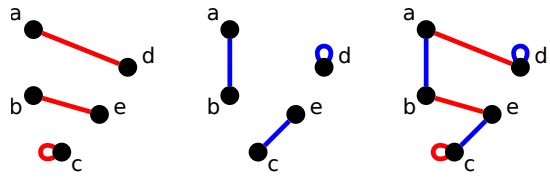


FIGURE 4.20 – *Représentation des involutions* $\{\alpha_1, \alpha_2\}$ *sur l'ensemble* $B = \{a, b, c, d, e\}$ *sous forme de graphes : (a)* $\alpha_1 = \{(a, d), (b, e), (c, c), (d, a), (e, b)\}$ *représenté par le graphe* (B, E_1) *; (b)* $\alpha_2 = \{(a, b), (b, a), (c, e), (d, d), (e, c)\}$ *représenté par le graphe* (B, E_2) *; (c)* (B, α_1, α_2) *représenté par le graphe* $(B, E_1 \cup E_2)$*. Des exemples similaires avec des permutations peuvent être consultés dans [6, Chap. 2.5.2].*

Définition 4.6.1 (Cartes généralisées : approche combinatoire)**.** *Soit* $n \geqslant 0$ *un entier naturel. Une n-carte généralisée, n-G-carte ou simplement G-carte, est un* $(n+2)$*-uplet* $M = (B, \alpha_0, \ldots, \alpha_n)$ *tel que :*

- *B est un ensemble fini de brins ;*
- $\alpha_0, \ldots, \alpha_n$ *sont des involutions sur B ;*
- $\alpha_i \circ \alpha_j$ *est une involution pour tout couple* (i, j) *de dimensions de* $0..n$ *tel que* $i + 2 \leqslant j$*;*

De manière générale, les modèles combinatoires utilisés pour la description d'objets géométriques reposent sur un ensemble de permutations ou d'involutions. Ces modèles admettent aussi une représentation par graphes : il suffit de superposer les graphes des involutions. Plus précisément, on considère chaque involution α_i comme une relation symétrique sur B, c'est-à-dire un sous-ensemble de $B \times B$. On peut alors traduire la structure $(B, \alpha_0, \ldots, \alpha_n)$ en un graphe avec B comme ensemble de nœuds et $E_1 \cup \ldots \cup E_n$ comme ensemble d'arcs. Chaque brin est alors considéré comme un nœud du graphe, tandis que chaque involution α_i produit un ensemble E_i d'arcs non orientés étiquetés par i et reliant ces nœuds. La figure 4.20 illustre cette construction. Les deux graphes 4.20a et 4.20b partagent $B = \{a, b, c, d, e\}$ pour l'ensemble des nœuds et possèdent un ensemble d'arcs déduits d'une involution, respectivement dessinés en rouge et en bleu. Le graphe final 4.20c a les mêmes nœuds et possède l'union des arcs des deux graphes comme ensemble d'arcs.

Par extension à l'ensemble des permutations permettant de décrire une carte généralisée, on obtient une définition sous forme de graphes des G-cartes. Notons qu'un graphe quelconque dont les arcs sont attribués ou typés par des dimensions n'encode pas nécessairement une G-carte. En

(a) Graphe type *(b) Graphe attribué type*

FIGURE 4.21 – *Graphes types pour les G-cartes et les G-cartes plongées*

effet, il faut aussi traduire la propriété que les fonctions sont des involutions et que certaines compositions sont à nouveaux des involutions. Nous avons donc recours à des contraintes, dites *topologiques*, qui assurent que le graphe représente une carte généralisée au sens de la définition 4.6.1.

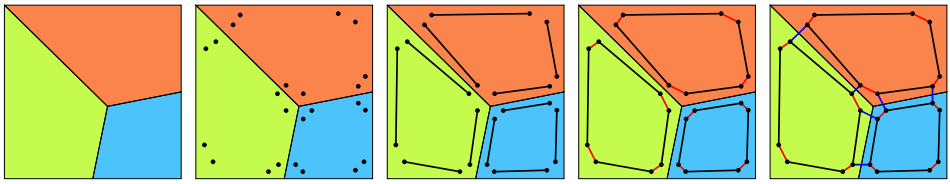
Définition 4.6.2 (Cartes généralisées : approche graphes)**.** *Un graphe non orienté $G = (B, E, i, \alpha)$ dont les arcs sont étiquetés sur $0..n$ est une carte généralisée de dimension n, n-G-map ou simplement G-carte, s'il satisfait les contraintes topologiques suivantes :*

Contrainte d'arcs incidents *pour tout nœud b de B, pour toute dimension d de $0..n$, il existe un unique arc $e \in E$ tel que e est incident à b, i.e., $i(e) = b$, et étiqueté par la dimension d, i.e., $\alpha(e) = d$.*

Contrainte de cycle *pour toutes les dimensions d et d' dans $0..n$ telles que $d + 2 \leqslant d'$, tout chemin de longueur 4 étiqueté par $dd'dd'$ est un cycle.*

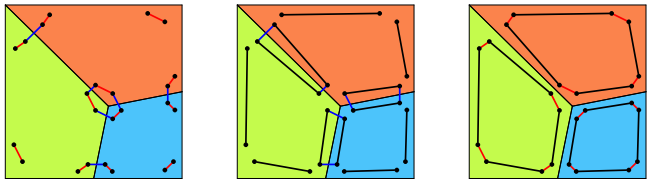
Décrite comme un graphe, une carte généralisée correspond à un graphe étiqueté sur les arcs par des dimensions (donc des valeurs de $\mathbb{N}$). En dimension 3, on peut donc la représenter comme un graphe typé, via les graphes types de la figure 4.21a. Ici encore la structure du graphe n'encode que la topologie de l'objet représenté et l'on a donc recours à des attributs pour ajouter des informations dites géométriques. Le graphes type de la figure 4.21a est ainsi étendu avec des plongements, notés génériquement π_i, possédant leur propre type de données, noté s_i. On obtient alors le graphe attribué type de la figure 4.21b.

Un brin dans une G-carte, i.e., un nœud du graphe, décrit un n-uplet de cellules topologiques. En dimension 2, un brin caractérise ainsi un triplet (sommet, arête, face), i.e. décrit un sommet, extrémité d'une arête, elle-même au bord d'une face. Les arcs du graphe encodent quant à eux les relations d'incidences et d'adjacences entre les cellules topologiques : un arc de dimension d entre deux brins indique que les cellules de dimension d des



(a) Objet *(b) Brins* *(c) 0-arcs* *(d) 1-arcs* *(e) 2-arcs*

FIGURE 4.22 – *Construction de la G-carte associée à un objet géométrique 2D*



(a) Sommets *(b) Arêtes* *(c) Faces*

FIGURE 4.23 – *Cellules topologiques*

deux brins sont adjacentes, avec la contrainte que ces deux brins doivent partager le même ensemble de cellules pour les autres dimensions. Par exemple, des arcs de dimension 2 séparent les faces de l'objet. On peut alors construire la carte généralisée encodant un objet géométrique, comme illustré en figure 4.22. L'objet de la figure 4.22a est en dimension 2, de sorte que les nœuds de la G-carte décrivent un triplet (sommet, arête, face). Les triplets valides sont donnés en figure 4.22b : chaque point représente un brin dessiné proche de son sommet, de son arête et de sa face. Il reste alors à spécifier les relations d'incidences et d'adjacences des cellules topologiques. On ajoute les 0-arcs (en noir) entre les brins d'une même arête et d'une même face mais de sommets différents, cf. figure 4.22c, puis les 1-arcs (en rouge) entre les brins d'arêtes différentes d'une même face et d'un même sommet, cf. figure 4.22d, et finalement les 2-arcs (en bleu) pour relier les faces, cf. figure 4.22e. Cette dernière figure représente la carte généralisée finale, sur laquelle nous n'avons pas représenté les 2-boucles du bord pour des soucis de lisibilité.

Le sens donné aux éléments (nœuds et arcs) d'une G-carte laisse entendre que les cellules topologiques de l'objet ne sont pas des parties atomiques du graphe mais des sous-graphes induits par un ensemble de dimensions. Par héritage de l'approche combinatoire, ces sous-graphes sont appelés *orbites*. Par exemple, en ne considérant que les dimensions 1 et 2, on n'autorise qu'à changer d'arête ou de face mais en restant toujours

dans le même sommet. Par suite, le sous-graphe contenant l'ensemble des nœuds atteignables depuis un nœud donné en n'empruntant que les 1-arcs et 2-arcs constitue donc le sommet associé au nœud, comme illustré en figure 4.23a. De la même manière, on représente les arêtes avec les orbites construites sur les dimensions 0 et 2 (figure 4.23b) et les faces avec les dimensions 0 et 1 (figure 4.23c). Il reste encore à compléter cette structure topologique avec des informations géométriques. Usuellement, on parle de plongement de sorte que l'on obtient un carte généralisée plongée. Cette notion est complètement encodée par l'attribution du graphe, de manière identique à la formalisation des maillages et nous ne la développerons pas plus.

4.6.3 Exemple inspiré de l'appareil de Golgi

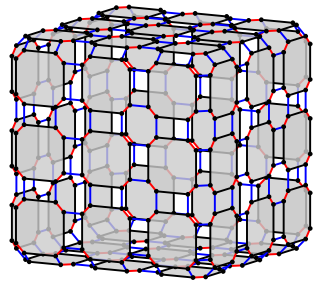
Nous avons maintenant l'ensemble des éléments nécessaires pour spécifier les différentes étapes de la modélisation de la création d'une vésicule, évoquée dans l'introduction de ce chapitre et illustrée en figure 4.1. Une telle représentation suppose de plonger la carte généralisée avec deux plongements a minima : la position des sommets et la concentration des protéines. Dans un souci de visualisation, nous avons aussi utilisé des couleurs sur les demi-faces, i.e. les orbites définies par les seules dimensions 0 et 1, les faces étant définies par les arêtes 0, 1 et 3 en dimension 3. L'objet d'origine est donné en figure 4.24a. Comme le graphe devient relativement conséquent, nous adoptons une description implicite des attributs : la localisation d'un nœud dans l'image est déduite de l'information de position du sommet auquel il appartient (calculée par explosion barycentrique) ; nous affichons les demi-faces comme des polygones définies par les orbites sur les dimensions 0 et 1, de sorte que la couleur associée à un nœud coïncide avec la couleur du polygone (ici toutes les demi-faces sont grises) ; nous n'affichons pas la concentration en protéines. À partir de la carte généralisée de la figure 4.24a, l'application de la règle donnée en figure 4.24b permet de réaliser l'opération de bourgeonnement par construction d'un nouveau volume recollé au volume d'origine le long d'une face. On obtient alors l'objet de la figure 4.24c. Après l'opération de migration qui consiste à augmenter la concentration en protéines du petit volume par transfert le long de la face commune, le petit volume est séparé via l'opération de la figure 4.24b. En plus de supprimer la relation d'adjacence des deux volumes, cette opération réalise la translation du petit volume et nous obtenons l'objet de la figure 4.24e.

Nous n'avons illustré que les éléments topologiques des règles car dans le cas présent, l'ajout d'attributs relatifs aux plongements géométriques alourdirait significativement l'écriture des règles. C'est pour cette raison que nous n'avons pas discuté dans cette section de l'étape de migration. Cette étape est purement géométrique et consiste à filtrer la face de recollement entre les deux volumes et à mettre à jour les concentrations dans le membre droit de la règle.

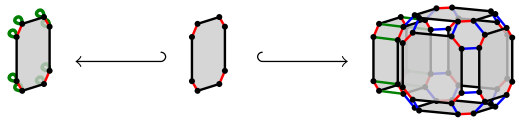
4.6.4 Extensions

Les techniques de transformations de graphes présentées dans ce chapitre permettent de modifier les structures de données utilisées en modélisation géométrique à base topologique. Elles présentent des avantages indéniables : format visuel et compact des règles, simplicité d'utilisation, préservation des attributs non mentionnés, garantie que le graphe modifié est bien formé au regard de la classe de graphes considérée (maillage, G-carte). Cependant, ces techniques restent trop conceptuelles et bas-niveau pour être utilisées telles quelles par un expert en modélisation géométrique souhaitant concevoir des opérations. Nous esquissons ci-dessous les briques essentielles à leur extension en vue de la construction d'un tel langage.

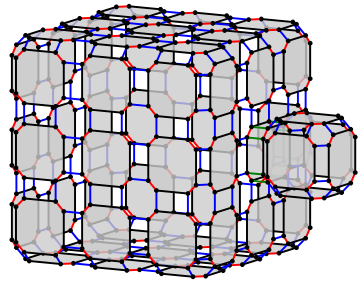
Extension topologique Les règles de transformation de graphes supposent que les membres gauche et droit de la règle encodent exactement le motif à modifier et le motif réécrit. Il est donc impossible d'écrire en une seule règle l'opération réalisant l'extrusion d'une face pour obtenir un prisme indépendamment du type de face (triangle, carré, pentagone…). Différentes approches permettent de généraliser une règle en un schéma de règles. Différentes modalités d'ajout de variables aux règles ont été étudiées dans [15]. Les instanciations admissibles des variables décrivent une famille de règles de transformations de graphes. Inspirés par ce travail, nous avons introduit des variables dites *topologiques* [25] qui permettent d'encoder les orbites pour définir la réécriture de G-cartes à orbite près. Une règle paramétrée par une variable *face* peut alors être déclinée en autant de règles de transformation de graphes qu'il y a de possibilités de construire des faces différentes. Par la suite, dans [23], nous avons reformulé les variables topologiques en termes catégoriels. La reformulation exploite les produits de graphes (cf. Définition 4.3.6) : intuitivement, les règles instanciées résultent d'un produit entre les graphes de la règle et d'un sous-graphe extrait du graphe à modifier.



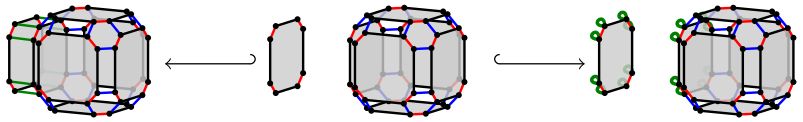
(a) Volume initial



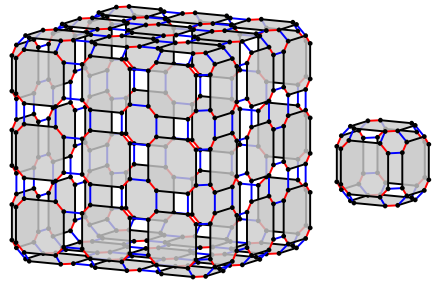
(b) Règle du bourgeonnement



(c) Volume bourgeonné



(d) Règle de détachement



(e) Vésicule détachée du volume

FIGURE 4.24 – *Règles de transformations de graphes pour la création d'une vésicule dans l'appareil de Golgi.*

Extension géométrique La construction de l'attribution des graphes que nous avons présentée ici, usuelle dans la communauté de transformations de graphes, permet l'ajout d'attributs aux éléments atomiques d'un graphe, i.e., à ses nœuds et à ses arcs. Cependant, en modélisation géométrique, les informations géométriques sont plutôt portées par les cellules topologiques de l'objet. Comme ces cellules sont encodées par des sous-graphes dans une carte généralisée, il convient d'adapter le cadre des transformations de graphes attribués pour prendre en compte cette particularité. Intuitivement il suffit d'encoder que certains éléments du graphe partagent la même valeur pour un attribut. Par ailleurs, la construction de l'attribution des graphes sépare complètement les manipulations algébriques des manipulations sur la structure du graphe, de sorte que les expressions algébriques sont indépendantes de la structure du graphe manipulé. Or, les calculs associés aux opérations de modélisation géométrique font très souvent appel aux éléments topologiques connexes aux attributs. Par exemple, les calculs géométriques peuvent dépendre de l'ensemble des couleurs des faces incidentes à un sommet, de l'ensemble des positions des sommets d'une face, ou encore de la couleur d'une face incidente. Nous avons enrichi les expressions de calculs géométriques avec des opérateurs dédiés qui permettent de collecter de façon générique les différents attributs indépendamment de la forme précise des composantes topologiques (faces, volumes, etc). En exploitant les éléments topologiques sous-jacents, ces opérateurs sont spécifiques aux G-cartes et enrichissent le pouvoir d'expression des expressions algébriques.

Préservation des contraintes définissant les modèles géométriques La formalisation d'objets géométriques sous la forme de graphes décorés permet la formalisation des opérations de modélisation géométrique sous la forme de règles de transformation de graphes. Cependant, nous avons vu que la définition sous forme de graphes des cartes généralisées ne peut être capturée comme une simple classe de graphes attribués typés. Nous avons donc recours à des contraintes caractérisant précisément la classe des cartes généralisées plongées comme une sous-classe de graphes attribués typés. Il s'agit maintenant de se demander s'il est possible d'énoncer des conditions sur les règles afin de garantir la préservation du modèle par application des règles. Plus généralement, la communauté de transformations de graphes s'est intéressée à cette question de préserver (ou de garantir) une propriété sur le graphe obtenu après transformation. Une formalisation en logique de premier ordre a été développée par K.-H. Pennemann [24] par le biais de conditions, connues sous le nom de *nested conditions*. Notons que cette

méthode ne fournit pas une solution statique indépendante de la règle, ce qui nous a amené à construire une solution *ad hoc* de vérification syntaxique des opérations tirant parti de la très forte régularité des cartes généralisées.

Graphes attribués typés vs modèles combinatoires Nous avons mis l'accent sur le modèle combinatoire des cartes généralisées. D'autres modèles utilisés en modélisation géométrique à base topologique reposent sur des formalisations similaires exploitant des permutations sur un ensemble d'éléments atomiques. Ces modèles admettent alors naturellement une représentation par graphes. C'est par exemple le cas des cartes combinatoires. Un lecteur attentif pourra se demander quel est l'avantage de raisonner dans la catégorie des graphes (décorés) plutôt que directement dans la catégories des cartes généralisées (ou d'un autre modèle combinatoire). La réponse est assez simple : il faudrait alors n'utiliser que des objets de la dite catégorie dans les règles. En relaxant la classe des graphes, à savoir en considérant une sur-classe de graphes attribués typés, les membres gauche et droit ainsi que l'interface des règles ne sont pas nécessairement des cartes généralisées mais simplement des graphes, ce qui laisse une plus grande liberté d'expression dans les règles.

Le mot de la fin Nous avons présenté un fragment de la théorie des transformations de graphes, sous un angle catégoriel, en insistant sur les formalisations possibles d'ajout d'information sur les graphes par le biais de décoration. Une telle approche fournit un cadre formel pour spécifier les transformations de systèmes dynamiques qu'il convient souvent de raffiner en fonction des besoins liés au domaine d'application, ce que nous avons esquissé dans le cadre de la modélisation géométrique.

Bibliographie

[1] M. BARR et C. WELLS : *Category theory for computing science*, vol. 49. Prentice Hall, New York, 1990.

[2] H. BELHAOUARI, A. ARNOULD, P. LE GALL et T. BELLET : Jerboa : A Graph Transformation Library for Topology-Based Geometric Modeling. *Actes de* H. GIESE et B. KÖNIG, éds : *Graph Transformation*, Lecture Notes in Computer Science, pp. 269–284, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-09108-2_18.

[3] A. J. COMPAORÉ : *Vers un environnement générique pour la prise en compte de la topologie des structures cellulaires dans les modèles de processus biologiques*. PhD Thesis, Evry-Val d'Essonne, mars 2012. URL `https://www.theses.fr/2012EVRY0003`.

[4] A. CORRADINI, D. DUVAL, R. ECHAHED, F. PROST et L. RIBEIRO : The PBPO graph transformation approach. *Journal of Logical and Algebraic Methods in Programming*, 103:213–231, fév. 2019. doi:10.1016/j.jlamp.2018.12.003. Publisher : Elsevier.

[5] A. CORRADINI, T. HEINDEL, F. HERMANN et B. KÖNIG : Sesqui-Pushout Rewriting. *Actes de* A. CORRADINI, H. EHRIG, U. MONTANARI, L. RIBEIRO et G. ROZENBERG, éds : *Graph Transformations*, Lecture Notes in Computer Science, pp. 30–45, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11841883_4.

[6] G. DAMIAND et P. LIENHARDT : *Combinatorial Maps : Efficient Data Structures for Computer Graphics and Image Processing*. CRC Press, sept. 2014.

[7] H. EHRIG : Introduction to the algebraic theory of graph grammars (a survey). *Actes de* V. CLAUS, H. EHRIG et G. ROZENBERG, éds : *Graph-Grammars and Their Application to Computer Science and Biology*, Lecture Notes in Computer Science, pp. 1–69, Berlin, Heidelberg, 1979. Springer. doi:10.1007/BFb0025714.

[8] H. EHRIG, K. EHRIG, U. PRANGE et G. TAENTZER : *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin Heidelberg, 2006. doi:10.1007/3-540-31188-2.

[9] H. EHRIG, M. KORFF et M. LÖWE : Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. *Actes de* H. EHRIG, H.-J. KREOWSKI et G. ROZENBERG, éds : *Graph Grammars and Their Application to Computer Science*, vol. 532 de *Lecture Notes in Computer Science*, pp. 24–37, Berlin, Heidelberg, 1991. Springer. doi:10.1007/BFb0017375.

[10] H. EHRIG, H.-J. KREOWSKI, U. MONTANARI et G. ROZENBERG : *Handbook of Graph Grammars and Computing by Graph Transformation : Volume 3 : Concurrency, Parallelism, and Distribution*, vol. Concurrency, Parallelism, and Distribution. World Scientific, USA, août 1999. doi:10.1142/4181.

[11] G. ENGELS : *Handbook of Graph Grammars and Computing by Graph Transformation : Volume 2 : Applications, Languages and Tools*, vol. Applications, Languages and Tools. World Scientific, USA, oct. 1999. doi:10.1142/4180.

[12] M. FISER, B. BENES, J. G. GALICIA, M. ABDUL-MASSIH, D. G. ALIAGA et V. KRS : Learning geometric graph grammars. *Actes de Proceedings of the 32nd Spring Conference on Computer Graphics*, SCCG '16, pp. 7–15, Slomenice, Slovakia, avr. 2016. Association for Computing Machinery. doi:10.1145/2948628.2948635. URL `https://doi.org/10.1145/2948628.2948635`.

[13] J.-L. GIAVITTO et O. MICHEL : MGS : A Rule-Based Programming Language for Complex Objects and Collections. *Electronic Notes in Theoretical Computer Science*, 59(4):286–304, nov. 2001. doi:10.1016/S1571-0661(04)00293-2.

[14] R. HECKEL et G. TAENTZER : *Graph Transformation for Software Engineers : With Applications to Model-Based Development and Domain-Specific Language Engineering*. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-43916-3. URL `http://link.springer.com/10.1007/978-3-030-43916-3`.

[15] B. HOFFMANN : Graph Transformation with Variables. *Dans* H.-J. KREOWSKI, U. MONTANARI, F. OREJAS, G. ROZENBERG et G. TAENTZER, éds : *Formal Methods in Software and Systems Modeling*, vol. 3393 de *Lecture Notes in Computer Science*, pp. 101–115. Springer, Berlin, Heidelberg, 2005. doi:10.1007/978-3-540-31847-7_6.

[16] S. IKEHATA, H. YANG et Y. FURUKAWA : Structured Indoor Modeling. *Actes de Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pp. 1323–1331, USA, 2015. IEEE Computer Society. doi:10.1109/ICCV.2015.156.

[17] B. KÖNIG, D. NOLTE, J. PADBERG et A. RENSINK : A Tutorial on Graph Transformation. *Dans* R. HECKEL et G. TAENTZER, éds : *Graph Transformation, Specifications, and Nets : In Memory of Hartmut Ehrig*, Lecture Notes in Computer Science, pp. 83–104. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-75396-6_5.

[18] S. LACK et P. SOBOCIŃSKI : Adhesive Categories. *Actes de* I. WALUKIEWICZ, éd. : *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, pp. 273–288, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-24727-2_20.

[19] T. LEINSTER : *Basic Category Theory*. déc. 2016. URL `http://arxiv.org/abs/1612.09375`. arXiv : 1612.09375.

[20] P. LIENHARDT : Subdivisions of N-dimensional Spaces and N-dimensional Generalized Maps. *Actes de Proceedings of the Fifth Annual Symposium on Computational Geometry*, SCG '89, pp. 228–236, New York, NY, USA, juin 1989. Association for Computing Machinery. doi:10.1145/73833.73859.

[21] M. LÖWE : Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109(1-2):181–224, mars 1993. doi:10.1016/0304-3975(93)90068-5.

[22] S. MAC LANE : *Categories for the working mathematician*, vol. 5 de *Grad. Texts Math.* Springer, New York, NY, 2nd ed édn, 1998. ISSN : 0072-5285.

[23] R. PASCUAL : *Inference of graph transformation rules for the design of geometric modeling operations*. PhD Thesis, Université Paris-Saclay, nov. 2022. URL `https://www.theses.fr/2022UPAST146`.

[24] K.-H. PENNEMANN : *Development of correct graph transformation systems*. PhD Thesis, Universität Oldenburg, 2009. URL `http://oops.uni-oldenburg.de/884/`.

[25] M. POUDRET : *Transformations de graphes pour les opérations topologiques en modélisation géométrique : application à l'étude de la dynamique de l'appareil de Golgi*. PhD Thesis, Evry-Val d'Essonne, oct. 2009. URL `https://www.theses.fr/2009EVRY0039`.

[26] M. POUDRET, J.-P. COMET, P. LE GALL, A. ARNOULD et P. MESEURE : Topology-based Geometric Modelling for Biological Cellular Processes. *Actes de LATA 2007. Proceedings of the 1st International Conference on Language and Automata Theory and Applications*, pp. 497–508, Tarragone, Spain, 2007.

[27] G. ROZENBERG, éd. *Handbook of Graph Grammars and Computing by Graph Transformation : Volume 1 : Foundations*, vol. Foundations. World Scientific, USA, fév. 1997. doi:10.1142/3303.

[28] A. SPICHER, O. MICHEL et J.-L. GIAVITTO : Algorithmic Self-assembly by Accretion and by Carving in MGS. *Actes de* E.-G. TALBI, P. LIARDET, P. COLLET, E. LUTTON et M. SCHOENAUER, éds : *Artificial Evolution*, Lecture Notes in Computer Science, pp. 189–200, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11740698_17.

[29] S. VILGERTSHOFER et A. BORRMANN : Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models. *Advanced Engineering Informatics*, 33:502–515, août 2017. doi:10.1016/j.aei.2017.07.003. Publisher : Elsevier.