Aide à la conception d'opérations topologiques par inférence

Romain Pascual [1], Hakim Belhaouari [2], Agnès Arnould [2] et Pascale Le Gall [1]

[1] Laboratoire Mathématiques et Informatique pour la Complexité et les Systèmes (MICS), CentraleSupélec, Université Paris Saclay
[2] Laboratoire XLIM UMR CNRS 7252, Université de Poitiers

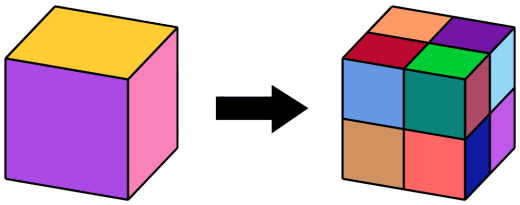
Résumé
La représentation sous forme de graphes des cartes généralisées permet d'étudier les opérations de modélisation comme des règles de transformation de graphes. En incorporant des mécanismes de variables typées par la nature des cellules topologiques concernées (volumes, faces, arêtes, ...), ces règles se généralisent à n'importe quelle cellule topologique du bon type. Par analyse syntaxique des règles, il est alors possible d'assurer la préservation des contraintes de cohérence du modèle. Ces règles étendues, appelées schémas de règles, sont notamment utilisées dans la plateforme Jerboa pour la conception de modeleurs basés sur le modèle des cartes généralisées. Cependant, la compréhension des schémas de règles suppose des connaissances pointues à fois dans le domaine des transformations de graphes et des cartes généralisées. Dans l'objectif de masquer ces éléments techniques, nous proposons un algorithme pour inférer des schémas de règles à partir de deux instances d'un même objet, avant et après modification, fournies par le concepteur de modeleurs.

Mots-clés : Modélisation topologique, Systèmes de fonctions itérées, Inférence d'opérations

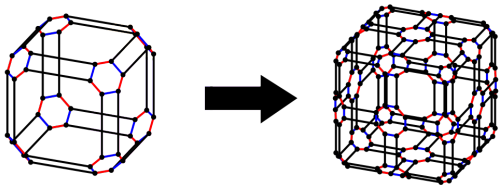
1. Introduction

Nos travaux s'inscrivent dans le domaine de la modélisation géométrique à base topologique [PB07]. Dans cet article, nous nous intéressons uniquement à la structure topologique de l'objet et laissons de côté les aspects géométriques comme le placement des sommets ou les autres informations de plongements portés par les *cellules topologiques* (volumes, faces, arêtes et sommets). En particulier, ce travail exploite le formalisme des *cartes généralisées* ou G-cartes [Lie89, Lie91], qui présentent l'avantage de se définir de manière homogène en toutes dimensions et d'offrir une définition sous forme de graphes [PCLG*07, BALG11]. À l'aide d'un tel formalisme, les opérations de modélisations se définissent comme des règles dans le cadre des transformations de graphes [Roz97, EEPT06, HT20]. Une étude formelle de ces règles permet d'assurer la préservation des contraintes topologiques des cartes généralisées [PACLG08] et celles liées aux calculs des plongements [BABLG17].

Intuitivement, une règle s'écrit $A \rightarrow B$ et permet de modifier l'objet A en l'objet B dans un contexte plus général. La subdivision du cube, illustrée en figure 1(a), s'exprime comme une règle où l'objet A est le cube avant subdivision et l'objet B est l'objet après subdivision. En représentant le cube sous forme de G-cartes, on en déduit la règle de transformation correspondante, donnée en figure 1(b). Cette règle permet de subdiviser n'importe quel parallélépipède, mais nous voudrions reconstruire l'opération de subdivision d'une surface quelconque.



(a) Application sur un cube.



(b) Règle de transformation de G-carte

Figure 1: *Opération de subdivision de quad*

L'idée est d'exploiter les cellules topologiques, et plus généralement les orbites, pour décrire comment modifier l'objet. Afin d'exprimer des transformations valables pour toutes les formes possibles d'un type d'orbite donné, les orbites deviennent des variables de la règle de transformation [PACLG08, BPA*10]. On obtient alors des schémas de règles qu'il est possible d'instancier en fonction de l'objet sur lequel on souhaite les appliquer. Ces schémas de règles sont au cœur de la plateforme Jerboa [BALGB14] (disponible en ligne à l'adresse `http://xlim-sic.labo.univ-poitiers.fr/jerboa/`). Jerboa permet la création d'opérations de modélisation : par génération de code, on obtient des opérations de modélisation prêtes à l'emploi pour un modeleur quelconque. Jerboa a été utilisé avec succès pour obtenir des modeleurs dédiés dans des domaines variés tels que l'architecture [CMS*19], la simulation d'évolution de plantes [BTG15] ou l'analyse de sol en géologie [GAB*16].

Même si notre formalisme permet de capturer des opérations topologiques complexes par des règles de transformation, ces règles s'avèrent très ésotériques, accessibles aux seules personnes avec une double expertise en modélisation géométrique et en transformations de graphes. Notre objectif est de proposer une méthode pour inférer des opérations de modélisation, en termes de transformations de graphes, à partir de la donnée d'une instance particulière de l'application de l'opération en cours de conception. Ainsi, l'expert en modélisation décrira l'opération de subdivision de quad par deux versions, avant et après modification, e.g. les cubes de la figure 1(a). Les opérations inférées seront correctes par construction et créées à la volée dans un éditeur intuitif pour l'utilisateur.

Dans cet article, nous présentons un algorithme permettant de reconstruire des règles de transformations à partir de deux cartes généralisées. L'objectif principal est d'identifier les cellules topologiques concernées par la transformation afin d'inférer une règle générique au regard de ces cellules. Notre approche sera décrite à l'aide d'exemples fil rouge, à savoir, les schémas de subdivisions, et principalement les schémas de quad (qui subdivisent les faces) et de Doo-Sabin [DS78] (qui subdivisent les sommets).

La section 2 rappelle le formalisme des cartes généralisées et des opérations de transformation. La section 3 présente l'algorithme de reconstruction des schémas de règles. Des résultats pour des schémas de subdivisions sont donnés en section 4. La section 5 explique les limites actuelles de notre approche et présente des perspectives d'amélioration.

2. Cartes généralisées

2.1. Structure

Ce travail utilise une définition des cartes généralisées (G-carte) sous forme de graphes.

Definition 2.1 (*Carte généralisée*)

Soit une dimension $n \geq 0$.

Une carte généralisée *de dimension n, G-carte de dimension n ou n-G-carte, $G = (V, E, s, t, \alpha)$ est un graphe étiqueté sur $[\![0, n]\!]$ vérifiant les contraintes de non-orientation, d'arcs incidents et de cycles. Autrement dit $G = (V, E, s, t, \alpha)$ est définie par :*

- *un ensemble de nœuds noté V, aussi appelés brins ;*
- *un ensemble d'arcs noté E, aussi appelés liaisons topologiques ;*
- *une fonction source $s : E \to V$ et une fonction cible $t : E \to V$ qui permettent de lier les arcs aux nœuds du graphe.*
- *$l : E \to [\![0, n]\!]$ est la fonction d'étiquetage, qui donne la dimension de chaque liaison.*
- *la* contrainte de non-orientation *: toute liaison e de E possède une liaison inverse $e' \in E$, i.e. $s(e') = t(e)$, $t(e') = s(e)$ et $l(e') = l(e)$.*
- *la* contrainte d'arcs incidents *: tout brin v de V est source (resp. cible) d'une unique i-liaison, pour tout i dans $[\![0, n]\!]$.*
- *la* contrainte de cycles *: tout brin v de V est source d'un $ijij$-cycle, pour tous i et j dans $[\![0, n]\!]$ tels que $i + 2 \leq j$.*

La représentation d'un objet par une G-carte peut se construire à partir de sa décomposition en cellules topologiques. Par exemple, l'objet 2D de la figure 2(a) peut être représenté par une 2-G-carte. L'objet est tout d'abord séparé en faces, liées le long de leur arête commune par un lien 2. Le lien signifie que les cellules de dimension 2 (les faces) partagent une arête. On obtient alors la figure 2(b). Cette décomposition est itérée par ordre décroissant des dimensions pour obtenir la figure 2(c) après séparation par la dimension 1 et finalement la figure 2(d) après la dimension 0. Les liens correspondent alors aux arcs de la G-carte, représentée en figure 2(e).

Dans la suite, on considère donnée une n-G-carte G. Les cellules topologiques (sommets, arêtes, faces, volumes) de l'objet géométrique représenté par G correspondent à des cas particuliers d'orbites. Une orbite de G est un sous-graphe contenant l'ensemble des brins atteignables par des liaisons appartenant à un sous-ensemble de $[\![0, n]\!]$ à partir d'un brin de G.

Definition 2.2 (*Orbite*)

Soit une dimension $n \geq 0$, $G = (V, E, s, t, \alpha)$ une n-G-carte et $\langle o \rangle$ un sous-ensemble de $[\![0, n]\!]$.

*Pour un nœud v de V, l'*orbite $\langle o \rangle$ *incidente à v, notée $\langle o \rangle(v)$ est l'ensemble des nœuds de V qui peuvent être atteints à partir de v en suivant un chemin étiqueté par une succession (éventuellement vide) d'éléments de $\langle o \rangle$. $\langle o \rangle(v)$ est dite de type $\langle o \rangle$ ou simplement qualifié de $\langle o \rangle$-orbite.*

Étant donné $\langle o \rangle(v)$ une orbite incidente au nœud v, on construit le graphe orbite incident à v, noté $G\langle o \rangle(v)$

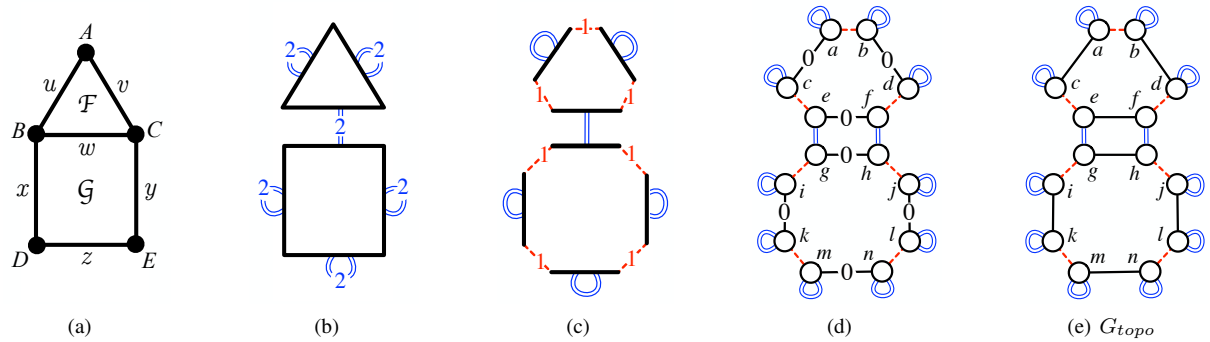


Figure 2: *Décomposition topologique d'un objet géométrique en dimension* 2.

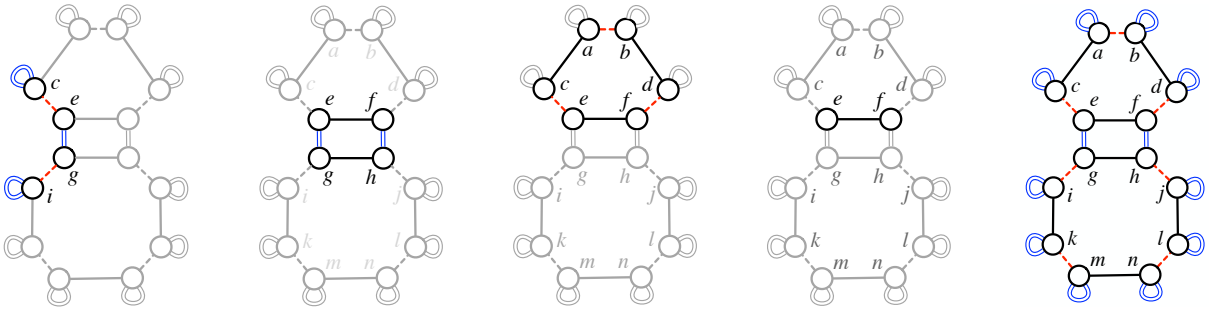


Figure 3: *Orbites incidentes à* e.

> *comme le sous-graphe induit par* $\langle o\rangle(v)$ *à l'aide des arcs étiquetés sur* $\langle o\rangle$. $G\langle o\rangle(v)$ *correspond donc à l'ensemble de nœuds* $\langle o\rangle$ *complété par les arcs utilisés pour relier les nœuds.*

Par exemple, la 0-cellule (sommet) incidente au brin e est représentée en figure 3(a). Cette cellule contient le brin e, et tous les brins atteignables en utilisant des liaisons de dimension 1 ou 2, ainsi que les liaisons elles-mêmes. La cellule correspond donc à une $\langle 1,2\rangle$-orbite. La figure 3(b) illustre la 1-cellule incidente au brin e, qui correspond à l'arête ω de la figure 2(a). Il s'agit du sous-graphe contenant les brins e, f, g et h et les liaisons les reliant. De même, la 2-cellule incidente à e est donnée en figure 3(c) et correspond à la face $\mathcal{F}$ de la figure 2(a).

2.2. Opérations de modélisation

La méthode de subdivision de Catmull-Clark utilisée pour le lissage de maillage quad est commune en informatique graphique [CC78]. Restreinte à la seule structure topologique (i.e. sans considération des aspects géométriques), nous nommons cette opération subdivision de quad. Un nouveau sommet est ajouté au centre de chacune des faces, ainsi qu'au milieu de chacune des arêtes. Le centre de chaque face est ensuite relié par une arête aux milieux de chacune de ses arêtes initiales. Appliquée sur un cube, chacune des faces est subdivisée en quatre nouvelles faces. On peut alors représenter l'opération de subdivision de quad d'un cube à l'aide de la règle décrivant le cube avant et après l'opération, comme sur la figure 1(a). La règle de subdivision de quad dans le formalisme de Jerboa (qui subdivise donc n'importe quelle surface) est donnée en figure 4.

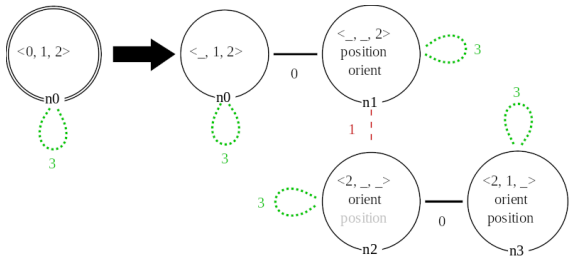


Figure 4: *Schéma de règles pour l'opération de subdivision de quad.*

Dans le schéma de règles de la figure 4, le motif gauche contient un unique nœud. Le double trait autour du nœud signifie qu'il s'agit d'une ancre, ou *hook*. Lors de l'application de la règle, l'ancre sera associée à un brin de l'objet, puis la partie de l'objet à modifier (le sous-graphe) sera filtré. Dans notre exemple, le nœud du motif gauche est étiqueté par $\langle 0,1,2\rangle$, c'est donc tout le volume du brin associé qui est filtré. Chaque nœud du schéma de règles représente ce sous-graphe ou l'une de ses copies à suppression et renommage près de ses liaisons. Par exemple, le nœud $n0$ apparaît à gauche et à droite du schéma de règles. Ainsi, le schéma préserve tous les brins filtrés par le nœud $n0$. Néanmoins, l'étiquette de $n0$ est modifiée. L'orbite $\langle 0,1,2\rangle$ de gauche est remplacée par $\langle \_,1,2\rangle$ à droite. Cela indique, en suivant les positions dans l'étiquette du nœud, que les liaisons 0 sont supprimées (noté par le souligné "_"), et les liaisons 1 et 2 préservées. Les nœuds $n1$, $n2$ et $n3$ n'apparaissent qu'à droite du schéma de règles. Ils correspondent donc à des créations et indiquent que les brins filtrés par l'ancre $n0$ sont dupliqués 3 fois. La copie $n1$ du volume filtré, supprime les liaisons 0 et 1, et ne conserve que les liaisons 2, comme l'indique son étiquette $\langle \_,\_,2\rangle$ comprenant deux "_". La copie $n2$, renomme toutes les liaisons 0 en liaisons 2 et supprime les liaisons 1 et 2, comme l'indique son étiquette $\langle 2,\_,\_\rangle$ où le 0 a été remplacé par un 2 et les 1 et 2 ont été remplacés par des "_". Enfin, le nœud $n3$ renomme les liaisons 0 en liaisons 2, préserve les liaisons 1 et supprime les liaisons 2. Les liaisons issues des étiquettes des nœuds, sont appelées liaisons implicites. À l'inverse, les arcs présents entre les nœuds du schéma de règles sont appelés liaisons explicites. Ils permettent de relier deux à deux les copies d'un même brin de la G-carte. Par exemple, le 0-arc entre les nœuds $n2$ et $n3$ signifie que chaque brin de la copie associée au nœud $n2$ est relié par une liaison 0 à son homologue dans la copie associée au nœud $n3$.

Les schémas de règles avec variables d'orbite sont un format intermédiaire essentiel pour la génération des opérations de modélisation géométrique dans le contexte de leur conception via la plateforme Jerboa. Cependant, leur écriture nécessite un apprentissage de ce langage spécifique et une bonne connaissance des cartes généralisés. Nous proposons donc de les inférer automatiquement à partir d'instances de la règle (d'un exemple d'objet avant et après l'application de l'opération à construire). Cette démarche dispensera le concepteur de comprendre le format de ces règles avec variables et permettra d'élaborer des schémas qui, par construction, satisferont les conditions syntaxiques de préservation de la consistance du modèle.

3. Inférence d'opérations

Nous avons deux niveaux d'abstraction pour les règles de transformation. Nous voudrions fournir à l'utilisateur un mécanisme permettant d'inférer une opération de modélisation générique à partir d'exemples. Autrement dit, on cherche à reconstruire le schéma de règles qui correspond à une modification de l'objet, réalisée par un utilisateur non-expert de la théorie sous-jacente.

On suppose qu'un utilisateur décrit un objet de départ et l'objet d'arrivée correspondant à l'application de l'opération recherchée. Notre objectif est alors d'inférer cette opération dans le formalisme de Jerboa pour une utilisation future sur différents objets.

Dans le cas d'une règle de création qui possède un motif gauche vide, la solution est triviale, puisque le seul schéma possible est la règle fournie par l'utilisateur. Cependant, dès lors que la règle possède un motif gauche non-vide, il existe plusieurs schémas de règles plausibles qui peuvent s'instancier en la règle fournie par l'utilisateur. Nous cherchons donc à reconstruire l'ensemble de ces schémas de règles puis à déterminer lequel correspond effectivement à l'opération de l'utilisateur. Pour ce faire, nous proposons d'appliquer les différents schémas de règles inférés sur des exemples générés automatiquement et de demander une discrimination par l'utilisateur pour identifier l'opération souhaitée. Nous avons donc besoin de deux étapes :

1. un mécanisme de reconstruction de schémas plausibles, i.e. possédant une instanciation égale à la règle donnée par l'utilisateur ;
2. un mécanisme de construction de G-cartes à partir d'un schéma de règles.

Nous appelons inférence d'opérations le premier mécanisme et inférence d'instanciations le second. Nous ne traiterons ici que de l'inférence d'opérations.

3.1. Notations

L'idée est de reconstruire l'ensemble des schémas qui permettent, via le mécanisme d'instanciation, d'obtenir une règle donnée. Plutôt que de reconstruire directement le schéma de règles, nous proposons un algorithme qui permet de reconstruire un schéma de graphes. Intuitivement, un schéma de graphes correspond au motif gauche (ou droit) d'un schéma de règles. Il s'agit donc d'un graphe où chaque nœud se voit associé une variable d'orbite. On expliquera ensuite comment modifier légèrement l'algorithme pour construire des schémas de règles.

Pour reconstruire un schéma de graphes associé à une G-carte, on commence par choisir un brin a dans la G-carte et un type d'orbite $\langle o\rangle$. On fait ensuite l'hypothèse que le graphe orbite incident au brin choisi correspond à l'instanciation de la variable d'orbite de l'ancre h dans le schéma de graphes $\mathcal{G}$. Par recherche locale, on essaie ensuite de reconstruire le schéma. Intuitivement la construction du schéma $\mathcal{G}$ consiste à replier le graphe G le long de ses i-liaisons, pour toutes les dimensions i du type d'orbite $\langle o\rangle$ choisi. Cette construction est renouvelée pour chaque couple brin et type d'orbite possible, appelée graine.

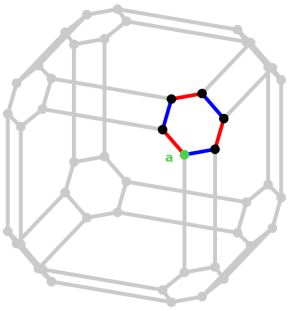


Figure 5: *Étape 1 - Construction de* $G\langle 1,2\rangle(a)$

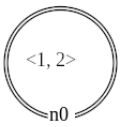
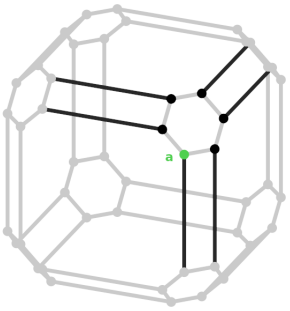
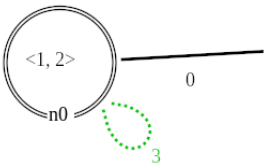


Figure 6: *Étape 2 - Construction de l'ancre*



(a)



(b)

Figure 7: *Etape 3 - Construction des arcs explicites de l'ancre*

3.2. Algorithme

Détaillons l'algorithme de construction d'un schéma pour une G-carte connexe G donnée et une graine choisie constituée d'un brin a de G et d'un type d'orbite $\langle o\rangle$. Si le schéma $\mathcal{G}$ existe, il aura une ancre h étiquetée par $\langle o\rangle$.

Dans la suite, nous illustrerons l'algorithme en utilisant le type d'orbite $\langle 1,2\rangle$ sur la G-carte d'un cube, autrement dit le membre gauche de la règle donnée en figure 1(b). Tous les brins sont incidents à une 3-boucle. Ces 3-boucles ne sont pas représentées pour ne pas surcharger l'image.

Étape 1 (Graphe orbite) On construit le graphe orbite $G\langle o\rangle(a)$ sur a dans G.

Le graphe orbite $G\langle 1,2\rangle(a)$ sur le cube est illustré en figure 5. Les liaisons et brins non-pertinents ont été grisés. Le graphe orbite comporte donc 6 brins et 3 doubles liaisons (chaque trait correspond à deux liaisons d'orientation inverse).

Étape 2 (Construction de l'ancre) On initialise le schéma par son ancre h étiquetée par le type d'orbite choisi $\langle o\rangle$. $G\langle o\rangle(a)$ est par construction une instanciation possible de ce schéma.

Dans le cas du cube, on obtient alors le graphe donné en figure 6 contenant une simple ancre $n0$ étiquetée $\langle 1,2\rangle$. Le sous-graphe de la figure 5 est bien l'une de ses instanciations possibles.

Étape 3 (Construction des arcs explicites de l'ancre) Les arcs explicites du schéma, incidents à l'ancre, sont étiquetés par les dimensions d qui n'appartiennent pas au type d'orbite $\langle o\rangle$. Chaque d-arc du schéma doit s'instancier en une d-liaison de G pour chaque brin de $G\langle o\rangle(a)$. Pour qu'une telle instanciation existe, deux cas sont possibles :

1. soit le d-arc du schéma est une boucle et tous les brins de $G\langle o\rangle(a)$ possèdent une d-boucle dans G ;
2. soit le d-arc du schéma relie l'ancre h à un autre nœud et tous les brins de $G\langle o\rangle(a)$ possèdent, dans G, une d-liaison vers des brins deux à deux distincts et extérieurs à $G\langle o\rangle(a)$.

Reprenons notre exemple. Comme nous l'avons dit précédemment, tous les brins de G, et donc de $G\langle 1,2\rangle(a)$ possèdent des 3-boucles (non représentées sur les figures). Donc l'ancre possède une 3-boucle dans le schéma en construction. De même, comme le montre la figure 7(a), les 6 brins du sommet $G\langle 1,2\rangle(a)$ possèdent une 0-liaison vers 6 brins distincts extérieurs au sommet filtré par l'ancre $n0$. On en déduit alors l'amorce de schéma de graphes donnée en figure 7(b) : ici, le schéma est incomplet puisque l'arc 0 pointe dans le vide.

Étape 4 (Construction d'un nœud) La construction d'un d-arc explicite suppose que les brins cibles des d-liaisons correspondantes soient l'instanciation d'un nouveau nœud m dans le schéma. Il convient de reconstruire ce nœud, en particulier ses arcs implicites. On cherche donc s'il existe une étiquette $\langle o^m\rangle$ dont l'instanciation fournit les liaisons qui joignent les brins de l'instance de m.

Cela revient à identifier les liaisons implicites préservées, supprimées et renommées par le nouveau nœud m. Pour tout i de $\langle o\rangle$:

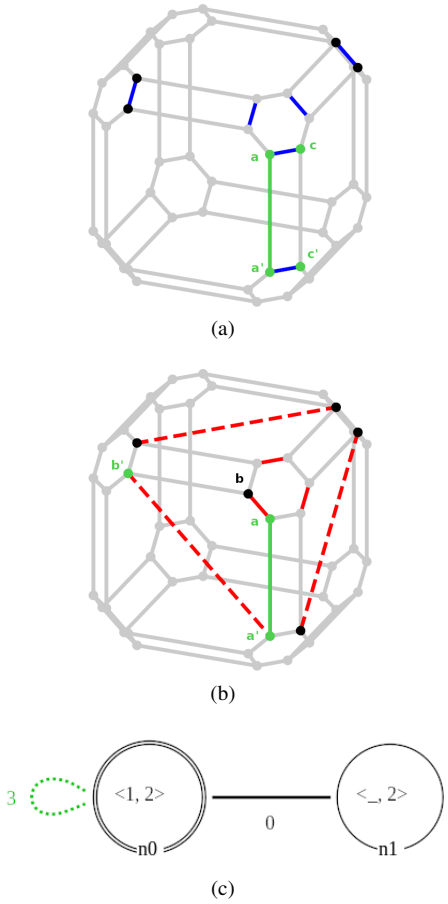


Figure 8: *Étape 4 - Construction des arcs implicites*

- si pour tout couple de brins i-liés, b et c de $G\langle o\rangle(a)$ (c'est-à-dire instance de l'ancre h), si leurs d-voisins respectifs, b' et c', sont i-liés, alors le nœud m préserve l'arc implicite i, et i apparaît dans $\langle o^m\rangle$ à la même position que dans $\langle o\rangle$;
- si pour tout couple de brins i-liés, b et c de $G\langle o\rangle(a)$, si leurs d-voisins respectifs, b' et c', sont j-liés, alors le nœud m renomme l'arc implicite i en j, et j apparaît dans $\langle o^m\rangle$ à la position de i dans $\langle o\rangle$;
- enfin, si pour tout couple de brins i-liés, b et c de $G\langle o\rangle(a)$, si leurs d-voisins respectifs, b' et c', ne sont pas liés, alors le nœud m supprime l'arc implicite i, et "_" apparaît dans $\langle o^m\rangle$ à la position de i dans $\langle o\rangle$.

Nous illustrons cette étape sur le 0-arc pendant de la figure 7(b) qui s'instancie en les 6 doubles 0-liaisons de la figure 7(a). Créons un nouveau nœud $n1$ dans le schéma et construisons son étiquette. On essaye de construire un renommage de 1 et 2 sur cet ensemble de brins. Comme illustré sur la figure 8(a), on obtient un renommage plausible de la 2-liaison entre a et c en une 2-liaison entre a' et c'. À l'inverse, comme illustré sur la figure 8(b), il n'existe pas de renommage de la 1-liaison entre a et b, cette liaison est supprimée. On vérifie que le renommage (le remplacement de 2 par 2 et la suppression de 1) est correcte vis-à-vis de l'ensemble des brins. Comme illustré sur la figure 8, ce renommage est valide. On en déduit alors le schéma de graphes partiel donné en figure 8(c).

Notons qu'en toute généralité, deux brins b et c de $G\langle o\rangle(a)$ peuvent être reliés par plusieurs i-liaisons implicites (i.e. avec différents i de $\langle o\rangle$). On peut alors construire plusieurs renommages différents. Notre algorithme utilise une heuristique pour construire les renommages plausibles parmi tous ceux possibles. Ainsi plusieurs schémas peuvent être construits pour une même graine de départ.

Étape 5 (Parcours) À l'aide d'un parcours en largeur de G, on reconstruit un schéma de graphes $\mathcal{G}$ qui donne G par instanciation. Autrement dit, le schéma de graphes permet de filtrer G à partir du brin a sur son ancre h.

Cela consiste à alterner les étapes 3 et 4 en les généralisant :

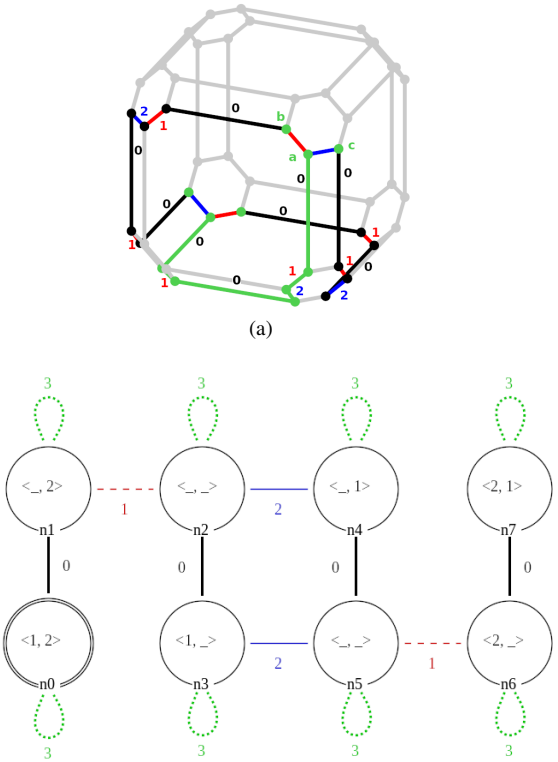
- Un troisième cas s'ajoute à la construction des arcs explicites, le cas où le d-arc relie le nœud en cours de traitement à un autre nœud déjà existant dans le schéma. La construction de l'arc explicite n'est alors suivie d'aucune création de nœud.
- Le renommage entre les arcs implicites $\langle o\rangle$ de l'ancre h et ceux $\langle o^m\rangle$ du nœud courant m n'est plus construit le long d'un d-arc explicite unique, car l'ancre h et le nœud m ne sont pas nécessairement des voisins immédiats. Le renommage est alors construit le long d'un chemin d'arcs explicites ω qui relie h à m. Ce même chemin ω de liaisons relie alors les brins considérés b et c instances de l'ancre h, à leurs ω-voisins b' et c' instances du nœud courant.

Si l'algorithme termine sans échec, on obtient un schéma de graphes $\mathcal{G}$ topologiquement correct pour la graine considérée.

Sur le cube, l'algorithme termine par la construction du dernier nœud $n7$ pour le schéma de graphes. Ce nœud correspond au sommet opposé au sommet initial, comme illustré sur la figure 9(a). En suivant le chemin d'exploration 012010, on atteint les brins du sommet opposé avec renommage de la variable d'orbite $\langle 1,2\rangle$ en $\langle 2,1\rangle$. Le schéma construit est donné en figure 9(b).

3.3. Généralisation à un schéma de règles

Pour inférer les schémas de règles cohérents avec une règle donnée, on propose de considérer les G-cartes (gauches et droites) comme faisant partie d'un même graphe. À partir des deux G-cartes L et R, on construit le graphe $\kappa(L,R)$ comme le graphe union de L et R, auquel on ajoute des arcs étiquetés par κ (un symbole particulier) et reliant les deux



(a)

(b)

Figure 9: *Fin de l'algorithme*

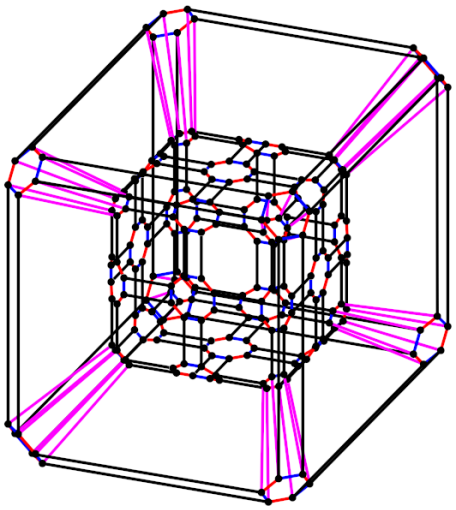


Figure 10: *Graphe $\kappa(L, R)$ pour la subdivision de quad du cube.*

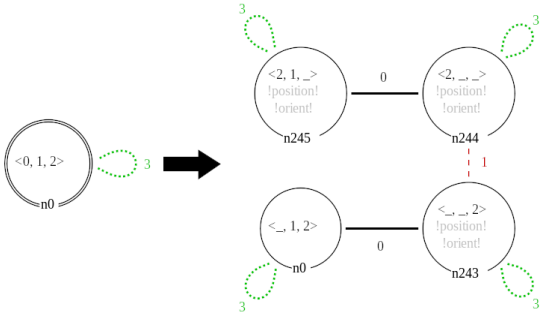


Figure 11: *Schéma inféré pour la subdivision de quad*

copies des nœuds préservés. Le graphe $\kappa(L, R)$ construit à partir de la règle de la figure 1(b) est donné en figure 10. Les κ-arcs sont représentés en rose.

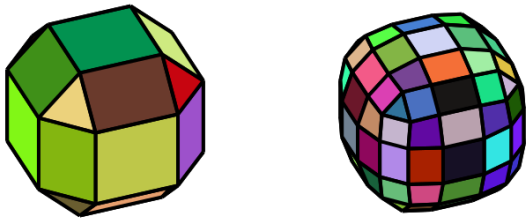
Pour déterminer un schéma d'une règle $r = L \twoheadrightarrow R$, on construit un schéma du graphe $\kappa(L, R)$ associé. On exécute l'algorithme sur le graphe $\kappa(L, R)$ en mémorisant si les brins appartenaient à L ou à R. La suppression des κ-arc du schéma de graphes produit le schéma de règles désiré. L'algorithme est identique sur le graphe $\kappa(L, R)$. Il convient simplement d'ajouter quelques conditions supplémentaires :

- on ne considère que les graines $(a, \langle o \rangle)$ pour lesquels a est un brin de L.
- à chaque identification d'une fonction de renommage de $\langle o \rangle$ vers $\langle o^m \rangle$, κ n'est pas une lettre de $\langle o^m \rangle$.

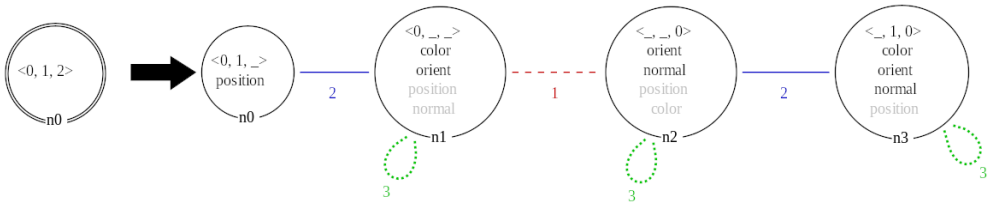
4. Exemples d'inférence sur des systèmes de fonctions itérées (IFS)

Nous avons reconstruit deux schémas de subdivision à l'aide de l'algorithme présenté. Le premier, illustré précédemment, est la subdivision de quad. Notons qu'un mécanisme de marquage des brins de la G-carte pendant le calcul de l'algorithme permet d'éviter d'explorer des graines pour lesquelles nous sommes certains d'obtenir un schéma déjà obtenu ou de ne pas aboutir à un repliement. En utilisant les deux cartes généralisées de la figure 1(b), nous obtenons 48 schémas valides. Parmi ces 48 schémas, nous obtenons 16 schémas distincts. Notons qu'il est effectivement possible de reconstruire 768 schémas (il y a 48 brins et 16 types d'orbite). La symétrie du cube assure qu'il n'y a en réalité qu'un schéma distinct par type d'orbite, soit 16 schémas. Le schéma de règles correspondant au repliement avec le type d'orbite $\langle 0, 1, 2 \rangle$ est donné en figure 11, il correspond au schéma présenté en figure 4 réalisé manuellement avec l'éditeur de règle de Jerboa.

La deuxième opération est la subdivision de surfaces Doo-Sabin [DS78]. Appliquée sur un cube, cette opération produit l'objet illustré en figure 12(a). Ici encore, la symétrie du cube assure que l'on obtient 48 schémas avec 16 schémas distincts parmi les 728 repliements possibles. Le schéma obtenu avec le type d'orbite $\langle 0, 1, 2 \rangle$ est donné en annexe A (figure 13) et similaire à celui utilisé (voir figure 12(c)).



(a) Application sur un cube. (b) Application itérée deux fois sur un cube.



(c) Schéma de règles utilisé.

Figure 12: *Opération de subdivision de Doo-Sabin.*

Évidemment, l'objectif de notre approche n'est pas d'obtenir des règles que nous avions déjà à notre disposition. Il s'agit plutôt de trouver des généralisations potentielles de nos opérations sur des orbites plus larges que celles initialement prévues dans la littérature. En effet, notre algorithme a montré qu'une opération pouvait se décliner sous plusieurs formes en fonction des repliements possible selon une graine de départ. Parmi elles, certaines sont dédiées à un usage très précis (volume isolé uniquement, structure topologique précise…) ou plus large (application de l'opération sur toute une composante connexe ou un volume). De plus, nous pouvons même aller plus loin en essayant de déterminer une opération comme étant une succession d'opérations. Ainsi, nous avons appliqué deux fois la subdivision de Doo-Sabin et inféré une opération réalisant directement deux itérations du schéma de subdivision. Sur un cube, on obtient alors l'objet illustré en figure 12(b). Le schéma de règles inféré avec le type d'orbite $\langle 0, 1, 2\rangle$ est illustré en annexe A (figure 14).

5. Conclusion et perspectives

L'algorithme présenté ici permet l'inférence de schémas de règles à partir de deux instances de cartes généralisées, avant et après modification. L'objectif est d'intégrer ce mécanisme dans un processus plus global d'apprentissage d'opérations de modélisation. La prochaine étape est d'obtenir un mécanisme d'inférence d'instances. A priori, nous envisagerions d'utiliser des stratégies provenant du test logiciel pour obtenir des graphes orbites. On pourrait ensuite utiliser ces graphes orbites pour obtenir une G-carte.

L'algorithme suppose que le graphe $\kappa(L, R)$ est connexe. Néanmoins, si ce graphe n'est pas connexe, l'opération que l'on cherche à inférer correspond à l'application parallèle de plusieurs opérations. Il nous semble donc acceptable de se limiter à l'inférence d'une opération à la fois dans un premier temps.

Une autre limite à notre algorithme est la prise en compte du contexte. En effet, nous construisons des schémas de règles à partir de deux cartes généralisées, avant et après modification. L'instanciation du motif gauche conduit donc forcément à une carte généralisée. Il n'est donc pas possible d'inférer des implémentations locales d'une opération, c'est-à-dire une opération qui ne filtre qu'une partie de la G-carte.

De plus, nous n'avons traité ici que la structure topologique des objets. Les schémas que nous inférons possèdent des annotations grisées qui signifient l'absence de spécification des valeurs de plongements. Il faudrait donc mettre au point une méthode pour inférer des calculs de plongements dans les schémas. L'évaluation de ces plongements pourraient être, eux aussi, déterminés par des mécaniques d'apprentissage et de discrimination par l'utilisateur sur des instances produites à la volée.

Références

[BABLG17] BELLET T., ARNOULD A., BELHAOUARI H., LE GALL P.: Geometric modeling: Consistency

preservation using two-layered variable substitutions. In *Graph Transformation (ICGT 2017)* (Cham, 2017), de Lara J., Plump D., (Eds.), Lecture Notes in Computer Science, Springer, pp. 36–53.

[BALG11] BELLET T., ARNOULD A., LE GALL P.: Rule-based transformations for geometric modeling. In *6th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2011)* (Saarbrücken, Germany, 2011), Echahed R., (Ed.), vol. 48, p. 20–37.

[BALGB14] BELHAOUARI H., ARNOULD A., LE GALL P., BELLET T.: Jerboa: A Graph Transformation Library for Topology-Based Geometric Modeling. In *Graph Transformation (ICGT 2014)* (Cham, 2014), Giese H., König B., (Eds.), vol. 8571 de *Lecture Notes in Computer Science*, Springer International Publishing, pp. 269–284.

[BPA*10] BELLET T., POUDRET M., ARNOULD A., FUCHS L., LE GALL P.: Designing a topological modeler kernel: a rule-based approach. In *Shape Modeling International Conference (SMI)* (2010), IEEE, pp. 100–112.

[BTG15] BOHL E., TERRAZ O., GHAZANFARPOUR D.: Modeling fruits and their internal structure using parametric 3gmap l-systems. *The Visual Computer. Vol. 31*, Num. 6 (2015), 819–829.

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design. Vol. 10*, Num. 6 (1978), 350 – 355.

[CMS*19] CARDOT A., MARCHEIX D., SKAPIN X., ARNOULD A., BELHAOUARI H.: Persistent naming based on graph transformation rules to reevaluate parametric specification. *Computer-Aided Design and Applications. Vol. 16*, Num. 5 (2019), 985–1002.

[DS78] DOO D., SABIN M.: Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design. Vol. 10*, Num. 6 (novembre 1978), 356–360.

[EEPT06] EHRIG H., EHRIG K., PRANGE U., TAENTZER G.: *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin Heidelberg, 2006.

[GAB*16] GAUTHIER V., ARNOULD A., BELHAOUARI H., HORNA S., PERRIN M., POUDRET M., RAINAUD J. F.: A topological approach for automated unstructured meshing of complex reservoir. European Association of Geoscientists & Engineers.

[HT20] HECKEL R., TAENTZER G.: *Graph Transformation for Software Engineers: With Applications to Model-Based Development and Domain-Specific Language Engineering*. Springer International Publishing, Cham, 2020.

[Lie89] LIENHARDT P.: Subdivisions of N-dimensional Spaces and N-dimensional Generalized Maps. In *Proceedings of the Fifth Annual Symposium on Computational Geometry* (New York, NY, USA, juin 1989), SCG '89, Association for Computing Machinery, pp. 228–236.

[Lie91] LIENHARDT P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design. Vol. 23*, Num. 11 (février 1991), 59–82.

[PACLG08] POUDRET M., ARNOULD A., COMET J.-P., LE GALL P.: Graph Transformation for Topology Modelling. In *Graph Transformations (ICGT 2008)* (Berlin, Heidelberg, 2008), Ehrig H., Heckel R., Rozenberg G., Taentzer G., (Eds.), vol. 5214 de *Lecture Notes in Computer Science*, Springer, pp. 147–161.

[PB07] PÉROCHE B., BECHMANN D.: *Informatique graphique, modélisation géométrique et animation*. Lavoisier, 2007.

[PCLG*07] POUDRET M., COMET J.-P., LE GALL P., ARNOULD A., MESEURE P.: Topology-based geometric modelling for biological cellular processes. In *International Conference on Language and Automata Theory and Applications* (2007), pp. 497–508.

[Roz97] ROZENBERG G. (Ed.): *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*, vol. Foundations. World Scientific Publishing Co., Inc., USA, février 1997.

Appendix A: Schémas inférés pour différentes itérations de la subdivision de Doo-Sabin

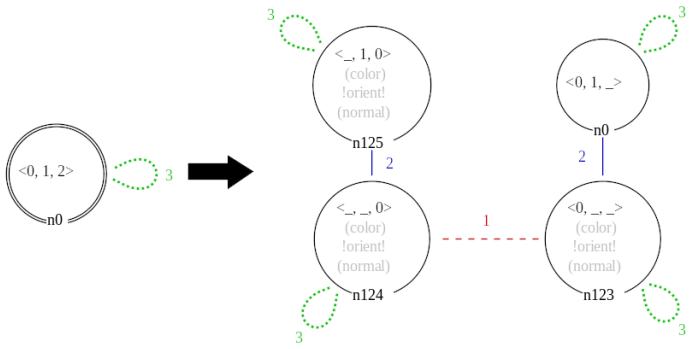


Figure 13: *Une itération.*

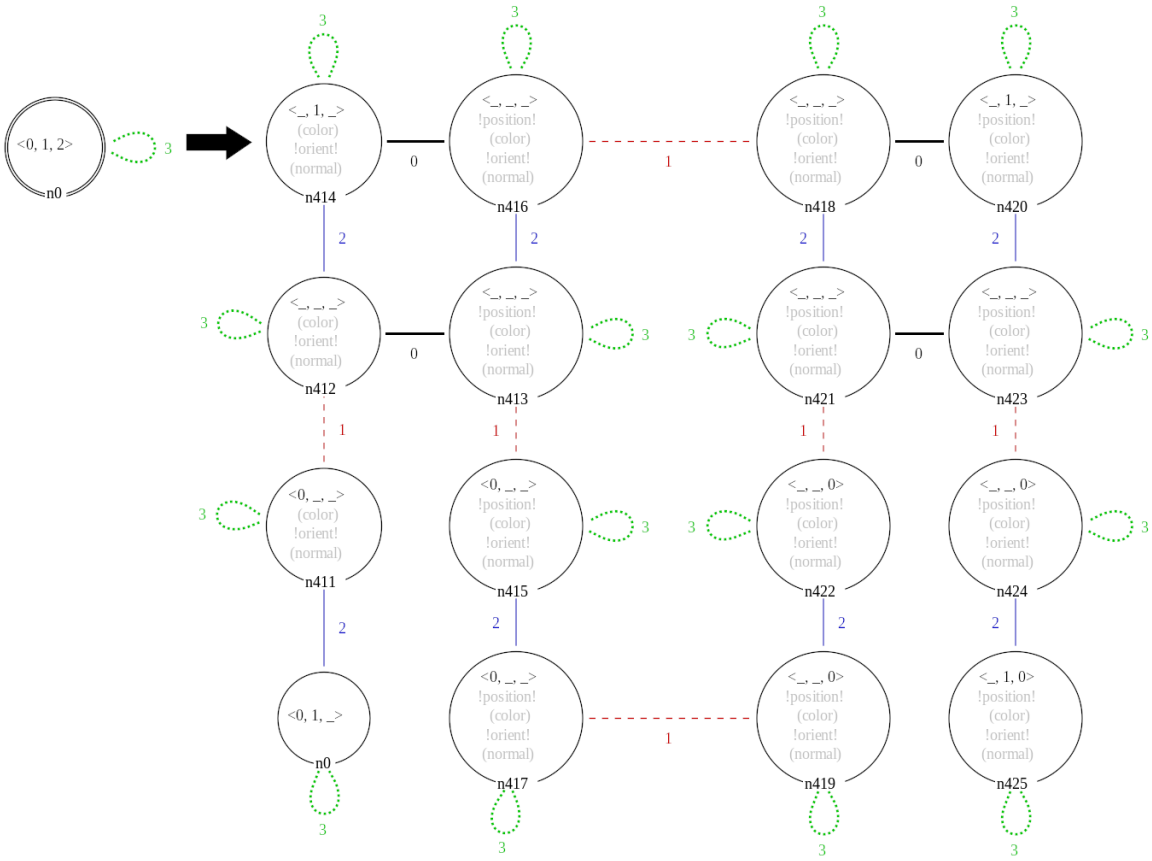


Figure 14: *Deux itérations.*