Rejeu basé sur des règles de transformation de graphes

Anaïs Cardot

THÈSE

pour l'obtention du Grade de

DOCTEUR DE L'UNIVERSITE DE POITIERS

(Faculté des Sciences Fondamentales et Appliquées)

(Diplôme National - Arrêté du 25 mai 2016)

Ecole Doctorale : Sciences et Ingéniérie des Systèmes, Mathématiques, Informatique

Secteur de Recherche : Informatique et Applications

Présentée par :

Anaïs Cardot

****

Rejeu basé sur des règles de transformation de graphes

****

Directeur de thèse : **Pascal Lienhardt**

Co-encadrants de thèse : **Xavier Skapin**, **David Marcheix**

****

Soutenance le 30 Janvier 2019

devant la Commission d'Examen

****

JURY

Président :

David Cazier	Professeur, Université de Strasbourg

Rapporteurs :

David Cazier	Professeur, Université de Strasbourg
Marc Daniel	Professeur, Ecole Polytech de Marseille

Membres du jury :

Guillaume Damiand	Directeur de Recherche, Université Claude Bernard, Lyon
Pascal Lienhardt	Professeur, Université de Poitiers
David Marcheix	Maître de Conférences, Université de Poitiers
Xavier Skapin	Maître de Conférences, Université de Poitiers

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse, Pascal Lienhardt, ainsi que mes co-encadrants, David Marcheix et Xavier Skapin pour leur accueil et leur travail de supervision.

Je remercie également les membres de mon jury de thèse, David Cazier, Marc Daniel et Guillaume Damiand, pour avoir relu ma thèse et m'avoir fait part de leurs remarques, mais également pour avoir été présents à ma soutenance malgré les conditions météorologiques difficiles.

Je remercie également les équipes du XLIM de Poitiers et du LIAS de l'ENSMA de m'avoir accueillie.

Je remercie Bénédicte, Caroline et Sophie, ainsi que François, pour leur accompagnement, leurs conseils et leur aide.

Je remercie mes amis : Anh-Toan, Armande, Aurélien, Camille, Caroline, Cédric, Corentin et Jorge, mais plus particulièrement Caroline et Sylvain, qui ont toujours été là, même dans les moments difficiles (Marty, on t'a attendu le 21 octobre 2015, pourquoi nous as-tu trahi ainsi ?).

Je remercie également mes deux BFF Forever, Mélanie, spécialiste de la protection des petits animaux en danger, et Christian alias "Jeff", qui a corrigé mon manuscrit en sa qualité d'ami prof de français suédois.

Je veux bien entendu remercier mes parents, Pascale et Roland, et ma sœur, Solèmne, qui m'ont encouragée, supportée et soutenue toutes ces années.

Enfin, parce qu'on n'est jamais à l'abri d'un oubli, je remercie toutes les personnes que j'ai oubliées.

Table des matières

So it begins

Théoden, The Lord of the Rings - The two Towers

1 Introduction

Sommaire

L'informatique graphique est apparue dans les années 60, d'abord pour la réalisation de plans dans l'industrie, puis s'est développée pour la création de pièces 3D par ordinateur (Catia, Solidworks...) et en parallèle, à partir des années 80, pour les jeux vidéo le cinéma (Cinéma 4D...). Depuis le début des années 2010, on assiste également au développement des réalités virtuelles et augmentées, permettant une meilleure immersion dans un logiciel

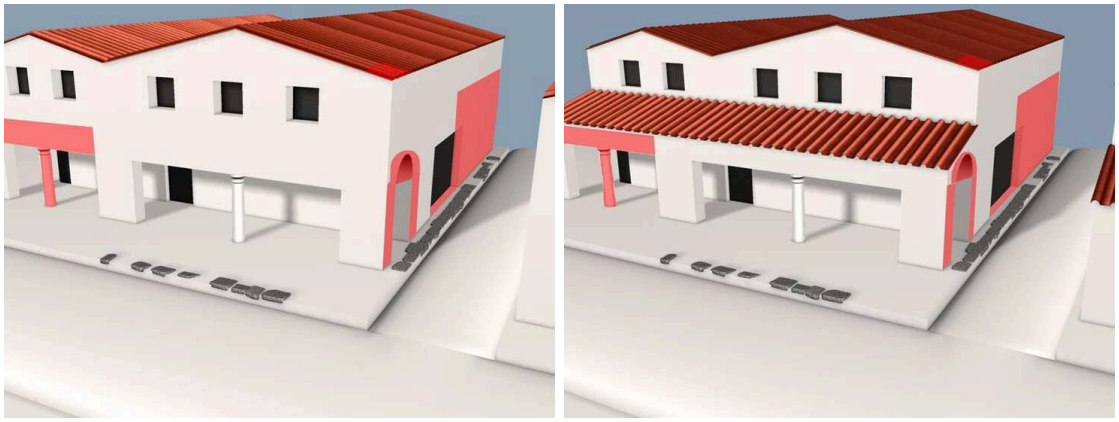
(peinture en 3D avec Tilt Brush par exemple), ou la pré-visualisation de travaux sur un site en construction grâce à une tablette pour l'architecture. La modélisation géométrique est également devenue un fantastique outil d'aide pour les archéologues, leur permettant de visualiser et de confronter plusieurs hypothèses, ou de reconstituer des sites historiques menacés de destruction, afin de continuer à en garder une trace.

1.1 Motivations

1.1.1 Contexte

Toutes ces utilisations de la 3D entraînent de nouveaux besoins et de nouvelles contraintes, comme l'obligation de réaliser des pièces pouvant exister dans la réalité pour la modélisation industrielle quand on les conçoit sur ordinateur, ou la nécessité de faire des modélisations les plus réalistes possible et donc les plus invisibles possible au cinéma.

Le besoin qui nous intéresse ici est celui de pouvoir réaliser plusieurs variations d'un même modèle sans que l'utilisateur n'ait à refaire entièrement le modèle en repartant de zéro à chaque fois. Ce besoin est particulièrement présent dans les domaines touchant à l'architecture et à la CAO (Conception Assistée par Ordinateur), afin de pouvoir réaliser plusieurs variations d'un bâtiment ou d'une pièce mécanique. On pourrait ainsi visualiser plusieurs variantes d'un même projet urbain, réaliser rapidement plusieurs bâtiments pour peupler un environnement vidéo ludique, ou encore, le domaine d'application qui nous intéresse le plus, réévaluer rapidement des hypothèses variées en archéologie. En effet, en archéologie, il peut ne rester que quelques traces au sol, parfois insuffisantes pour se faire une idée précise de la disposition du bâtiment. Il peut donc y avoir beaucoup d'incertitudes et d'hypothèses faites par les archéologues lorsqu'ils se trouvent face à des vestiges archéologiques. Prenons l'exemple de la figure 1.1. On peut y apercevoir deux configurations différentes pour un même bâtiment, avec une façade sur un seul plan pour la figure (a), et une façade renfoncée au 1er étage pour la figure (b). Les archéologues peuvent ainsi être amenés à réaliser plusieurs modèles afin de déterminer la configuration la plus probable d'un bâtiment par exemple.



(a) **(b)**

Figure 1.1 : Multiples hypothèses en archéologie, images fournies par l'HeRMA, réalisées par Mathieu Linlaud

$1 - ConstructionMaison(dimension1, dimension2, dimension3)$

(a)

$1 - ConstructionMaison(dimension1, dimension2, dimension3)$;
$2 - Renfoncement(profondeur, hauteur, faceAvant)$;
$3 - Pergola(angle, faceRenfoncement)$

(b)

Figure 1.2 : Multiples hypothèses en archéologie. a : Spécification du modèle de la figure 1.1a, avec les dimensions de la maison. b : Spécification du modèle de la figure 1.1b, avec $hauteur$ la hauteur sur laquelle on effectue le renfoncement, $faceAvant$ la face avant de la maison, et $faceRenfoncement$, la face créée par le renfoncement

1.1.2 Problématique

La reconstitution manuelle de toutes ces variations peut être fastidieuse, surtout quand il n'y a pas d'autre choix que de reconstruire le modèle en partant de zéro car il est impossible ou plus complexe de modifier le modèle existant pour obtenir le résultat voulu. De plus, l'utilisateur est obligé d'être présent pendant tout le processus de reconstruction.

Afin de pallier cela, nous utilisons un système de modélisation appelé *système paramétrique*, ou *modèle paramétrique* dont l'un des éléments, la *spécification paramétrique*, permet de stocker l'intégralité des opérations impliquées dans la construction d'un modèle, et donc d'éditer la liste des opérations et de changer leurs paramètres. Par rapport aux

modèles de la figure 1.1, on aura donc les spécifications paramétriques de la figure 1.2.

Afin de réaliser une modélisation paramétrique, il est également nécessaire de pouvoir identifier les entités de manière permanente. En effet, si l'on se contente par exemple d'identifier $faceAvant$ par un entier incrémenté, alors il suffit que la face soit créée avec un identifiant différent pour qu'on ne puisse plus l'identifier.

C'est ici qu'intervient la notion de *nommage persistant*. Celui-ci permet de caractériser de façon unique et non ambiguë une entité dans une première spécification, afin de pouvoir identifier la ou les entités correspondantes après avoir fait varier le modèle. Le fait de faire varier un modèle (les opérations constituants ce modèle et / ou les paramètres de ces opérations) pour ensuite le réévaluer est appelé le *rejeu*. Ici, par exemple, le nommage persistant nous permettrait de savoir que, suite au renfoncement, $faceAvant$ correspond maintenant à deux faces. Si sur la figure 1.1a, on colore cette face avant, quand on rejoue notre modèle on peut identifier les deux nouvelles faces créées par l'ajout du renfoncement (celle du rez-de-chaussée et celle du premier étage) et les colorer correctement.

1.1.3 Objectifs

Pour répondre à toutes ces problématiques, notre thèse a plusieurs objectifs :

Tout d'abord, réaliser un système robuste d'appariement, permettant non seulement l'identification des entités de manière formelle, mais aussi l'édition de spécification paramétrique (c'est-à-dire, l'ajout, la suppression, et le déplacement d'opérations), sujet rarement abordé dans la littérature existante. C'est cette édition qui permet d'ajouter un renfoncement sur la figure 1.1b.

De plus, nous voulons créer un système de nommage à la fois général, c'est-à-dire ne dépendant pas de la dimension du modèle, et homogène, donc ne dépendant pas de la dimension de l'entité nommée. Pour cela, nous nous basons sur le système de modélisation des *cartes généralisées* [Lie94], et sur les *règles de transformation de graphe* de la bibliothèque Jerboa [dP CP].

1.2 Trame du mémoire

Ce mémoire s'articulera autour de trois axes, tout d'abord un état de l'art de la modélisation et du nommage, puis une présentation de notre méthode, avant de comparer cette dernière aux méthodes pré-existantes.

1.2.1 État de l'art

Dans cette première partie, composée du chapitre 2, nous étudions les différentes méthodes de modélisation paramétrique et de nommage pre-existantes.

Pour cela, nous réalisons tout d'abord un bref historique de la modélisation assistée par ordinateur.

Dans un deuxième temps, nous réalisons un inventaire de différentes manières de représenter un modèle géométrique, en nous attardant sur celui que nous utilisons, le modèle topologique des cartes généralisées.

Dans un troisième temps, nous expliquons le concept de modélisation paramétrique, ainsi que les deux approches existantes, équationnelle et fonctionnelle, cette dernière étant celle que nous utilisons.

Dans un quatrième temps, nous abordons le concept de nommage persistant, en expliquant en quoi il est nécessaire, puis nous détaillons certains concepts communs à tous les types de nommage, avant de détailler plusieurs méthodes existantes de nommage persistant.

Enfin, nous expliquons le concept des règles de transformation de graphes et, plus particulièrement, celles que nous utilisons, les règles de la bibliothèque Jerboa, permettant de transformer des cartes généralisées.

1.2.2 Notre méthode de nommage, outils et application

Dans cette deuxième partie, composée des chapitres 3 et 4, nous allons expliquer notre méthode, en nous aidant d'un exemple fil rouge.

Dans le chapitre 3, nous développons les différents outils qui nous permettent de rejouer un modèle. Il y a tout d'abord le nommage persistant, basé sur le modèle des cartes généralisées. Nous détaillons ensuite notre système de journaux de bord, permettant de suivre les modifications apportées au modèle par une règle Jerboa. Enfin, nous présentons notre système de journaux d'historique, permettant, après avoir réalisé une spécification paramétrique initiale, de garder une trace des opérations ayant affecté l'entité, afin de pouvoir réaliser le rejeu.

Dans le chapitre 4, nous commençons par détailler les différents types d'édition pouvant affecter une spécification, à savoir l'ajout, la suppression, ou le déplacement d'opérations, ainsi que la modification de paramètres. Nous pouvons ensuite nous appuyer sur ces dif-

férents types et sur les outils du chapitre précédent pour créer nos arbres d'appariement, permettant, pour toutes les entités utilisées par la spécification paramétrique, de trouver leur(s) entité(s) correspondante(s) au rejeu. Enfin, nous appliquons tout ces éléments au rejeu de notre exemple fil rouge.

1.2.3 Comparaison aux autres méthodes

Dans cette troisième et dernière partie, nous comparons notre méthode aux autres méthodes de nommage existantes. Nous étudions d'abord la manière dont notre méthode s'inscrit dans les concepts communs, par rapport aux autres méthodes. Dans un deuxième temps, nous calculons, grâce à un exemple fil rouge, la taille de notre nommage et celle d'autres méthodes, puis, nous ferons la même chose avec la taille des mécanismes nécessaires à la réalisation de l'appariement.

Enfin, nous présentons une conclusion générale, et ainsi que différentes perspectives.

There's an endless road to rediscover

Avicii

2

État de l'art

Sommaire

L'informatique graphique, depuis les années 50-60, a énormément évolué en fonction des usages qui en étaient fait (conception de pièce, création d'environnements réalistes), et de domaines d'applications qui l'utilisent (industrie, cinéma....)

C'est pourquoi nous allons commencer cette partie par un bref rappel de l'évolution de l'informatique graphique. Nous détaillerons ensuite différentes manières de représenter un modèle géométrique, dont celle que nous utiliserons pour nos travaux, le modèle topologique. Dans un troisième temps, nous détaillerons le modèle permettant de modifier une modélisation après conception, en modifiant les opérations qui l'ont créé, le modèle paramétrique. En quatrième, nous étudierons le nommage persistant, un outil nécessaire à la modification d'un modèle par la modification de ses opérations constituantes. Enfin, nous expliquerons le fonctionnement des règles Jerboa, le système de réécriture de graphe que nous utiliserons pour nos travaux.

2.1 Rapide historique de l'informatique graphique

Comme beaucoup d'autres domaines, l'informatique graphique a évolué au cours des décennies, afin de s'adapter aux différents usages qui en sont faits (conception mécanique, jeux vidéo, architecture...), et de satisfaire leurs besoins spécifiques (imprimer des pièces, générer de grands espaces immersifs, simuler de nouveaux ameublements en réalité augmentée…).

Dans les années 50 et au début des années 60, différents outils commencent à apparaître pour représenter les objets réels, et forment les bases de ce qui deviendra la *CAO* (Conception Assistée par Ordinateur, ou Computer-Aided Design en anglais).

Parmi ceux-ci, on trouve :

— Les *machines à commande numériques*, qui permettent, au lieu de faire usiner une pièce par un artisan, d'entrer directement un programme informatique dans la machine afin que celle-ci l'utilise pour reproduire la pièce indiquée par le programme. Ce type de machine a un grand intérêt pour les pièces produites à grande échelle, car elles seront toutes identiques et usinées rapidement. La commande était à la

base passée à la machine sous forme de cartes perforées, mais est désormais passée directement grâce à un modèle de la pièce conçu sur ordinateur.

— Les *courbes et surfaces paramétriques*, permettant de décrire la représentation d'une géométrie utilisée à l'origine pour décrire l'aspect de pièces à usiner. En effet, les premiers exemples de courbes paramétriques sont utilisées pour décrire des pièces en aéronautique et en automobile, avec notamment, les travaux de Bézier pour Renault [Bé86], datant du début des années 60, ceux de Ferguson pour Boeing [Fer86], ou encore Casteljeau, qui a développé un algorithme d'approximation des polynômes pour Citroën [Boe93]. Il est à noter que, en raison du contexte de développement de ces découvertes, il peut être difficile de connaître leurs dates de début d'utilisation exacte, à cause du secret industriel.

— L'*infographie*, permettant à l'origine la réalisation de dessins en deux dimensions directement dans l'ordinateur. En 1963, Sutherland [Sut63] propose un système interactif de dessin 2D. Son logiciel, *Sketchpad*, permet de communiquer avec l'ordinateur *via* des dessins en deux dimensions. Avant cela, les actions comme par exemple décrire la forme d'une pièce mécanique se faisaient entièrement sur papier. À ce titre, il est considéré aujourd'hui comme le précurseur des logiciels de CAO actuels.

Benest [Ben79] propose un historique des systèmes d'informatique graphique, de Sketchpad au milieu des années 70. Il cite notamment une méthode développée en 1964 par General Motors de Dessin Assisté par Ordinateur, développée parallèlement à celle de Sutherland [Jac64], dans laquelle l'utilisateur communique avec l'ordinateur en utilisant un stylet et un panneau de verre situé en face de l'écran. Le stylet permettant d'indiquer une position sur l'ordinateur. Il indique qu'en 1966, IBM met en place un système permettant de dessiner par des moyens informatiques un circuit électronique, puis d'entrer grâce au clavier des informations par rapport à ces composants. Grâce au circuit et à ses information, le système peut ensuite calculer la résistance de chaque transistor [KSSH66]. En 1969, on commence à s'intéresser à l'animation assistée par ordinateur, que ce soit pour les dessins animés [Bae69], ou pour l'animation d'éléments industriels [The69]. Enfin, en 1976, une méthode permettant de réaliser une image médicale en 3 dimensions à partir de radiographies de coupes successives est mise au point [MH76].

Dans les années 70, on assiste également à l'apparition des *méthodes d'éléments finis* (MEF). Elle permettent, *via* des logiciels comme par exemple *Abaqus*, [Sysa], de simuler de manière virtuelle le comportement d'un objet et ainsi de réaliser des tests, sans risquer

d'endommager un véritable prototype.

Des années 80 à nos jours apparaissent de nombreux modeleurs, que ce soit pour l'industrie (*Catia* [Sysb], *Pro/Engineer* [Cor]), pour l'architecture, afin de prévisualiser de futurs bâtiments ou pour en reconstruire d'anciens, ou encore pour les jeux vidéos et le cinéma (*Blender* [Ble], *Cinema 4D* [Com], *Houdini* [Inc]...), qui développent de plus en plus leurs environnements 3D depuis la fin des années 90.

2.2 Modélisation géométrique

La modélisation géométrique est un ensemble de méthodes, algorithmes... permettant de réaliser des modèles informatiques d'éléments, grâce à des *plongements* (des informations associées aux éléments du modèle) et à de la *topologie* (domaine des mathématiques qui s'intéresse aux propriétés d'un espace et à sa préservation suite à des déformations). Ainsi, on peut réaliser, en 2 dimensions, des dessins techniques, des dessins artistiques, utilisés pour la reproduction à grande échelle, et en 3 dimensions, des pièces à usiner (certains logiciels, comme Catia, permettent de réaliser un dessin technique directement *via* un modèle 3D), des prototypes de mécanismes pour voir comment agencer les pièces en architecture, des environnements pour le cinéma ou les jeux vidéo…

2.2.1 Modèles classiques

2.2.1.1 Modèle de représentation par la construction géométrique

Le modèle *Constructive Solid Geometry* ou *CSG* consiste à créer un objet complexe à partir de plusieurs formes simples. Il a été développé par Requicha en 1977 [Req77].

Un objet modélisé en CSG est ainsi le résultat de l'application d'opérations booléennes sur des primitives 3D. Un objet est ainsi représenté *via* un arbre, dont les branches sont les opérations booléennes, et dont les feuilles sont les primitives.

La figure 2.1 représente un objet créé grâce au modèle CSG. La branche de gauche est composée de l'intersection d'un cube et d'une sphère. Celle de droite est composée de l'union d'un cylindre et de l'union d'un cylindre et d'un autre cylindre. L'objet final est la différence de ces deux branches.

Ce modèle est intéressant par rapport à notre problématique, en cela que l'on peut

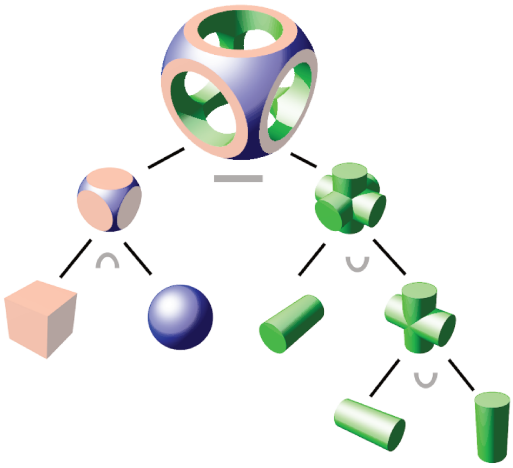


Figure 2.1 : Décomposition d'un objet selon le modèle CSG

modifier le modèle en changeant l'une des primitives ou l'une des opérations booléennes afin de réaliser une *variation* du modèle, c'est-à-dire, de créer une nouvelle version de celui-ci.

Cependant, le nombre de primitives peut alourdir le modèle, et augmenter le temps de rendu, car l'ordinateur doit toutes les prendre en compte. De plus, il n'y a pas unicité de la représentation (*ambiguïté*), un même objet pouvant être défini de plusieurs manières différentes. Enfin, la frontière de l'objet n'est pas définie explicitement, car elle résulte de l'interaction des primitives. On ne peut donc pas sélectionner spécifiquement un élément de ce bord.

2.2.1.2 Modèle de représentation par les frontières

Le modèle *Boundary Representation* ou B-Rep est développé au début des années 70 indépendamment par Braid [Bra75] et Baumgart [Bau75]. Il consiste à représenter un modèle par ses bords, en décrivant ces derniers. Ces bords sont orientés.

Ainsi, un cube sera décrit par les faces qui composent ses bords, un pentagone par les arêtes qui composent ses bords, et une arête par les sommets qui composent ses bords. La figure 2.2 présente la décomposition successive d'un cube *via* les bords des éléments qui le composent. Ainsi, le volume se décompose en 6 faces, chacune se décomposant en 4 arêtes, qui se décomposent elles-mêmes en 2 sommets.

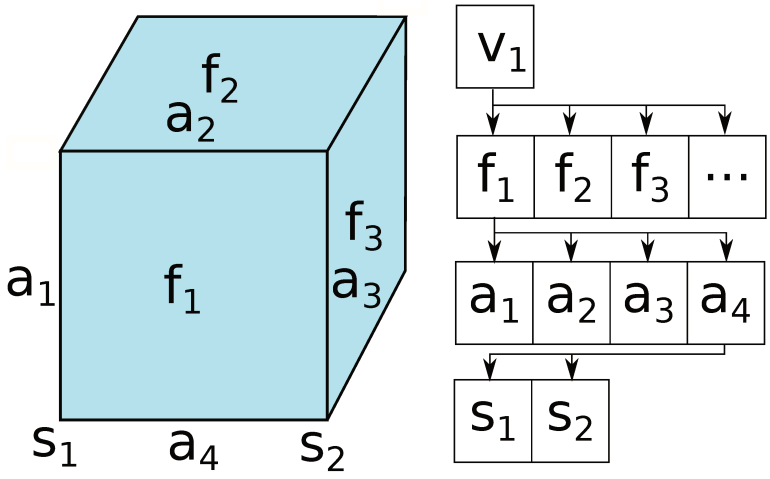


Figure 2.2 : Décomposition d'un objet selon le modèle BRep

L'intégrité de ces modèles peut être assurée en vérifiant qu'aucun de leurs éléments n'est ouvert, c'est-à-dire, que leurs bords ne sont pas disjoints.

On peut citer le modèle dit *facétisé* décrit par [Man88], dans lequel des coordonnées dans l'espace, qui permettent de situer les arêtes, les faces et les volumes, sont associées aux sommets. Cette information est associée au sommet grâce à une application appelée *plongement*.

2.2.2 Courbes et surfaces paramétriques

Les courbes et surface paramétriques [Gal99] sont étudiées depuis les années 60-70 environ, *via* notamment les travaux de Bézier [Bé86], de Ferguson [Fer86] et de Casteljau [Boe93, DC85].

Dans ce modèle, les éléments géométriques sont définis *via* un ensemble de points, appelés *points de contrôle* ou *pôles*. Une courbe paramétrique est alors mathématiquement définie comme

$$C(t) = \sum_{i=0}^{n} \phi_i(t) * P_i$$

avec t un paramètre, P_i les pôles successifs, et ϕ_i les fonctions successives. On peut en voir un exemple sur la figure 2.3.

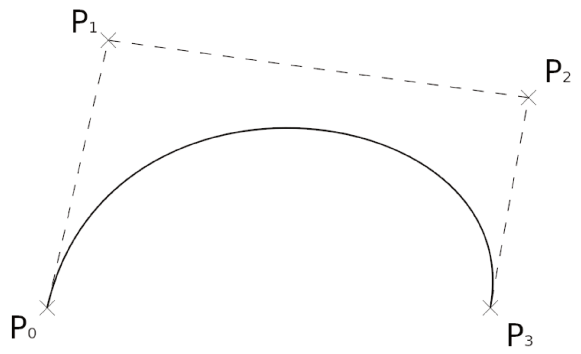


Figure 2.3 : Courbe de Bézier à 4 pôles

Le choix de ces ϕ_i permet de différencier les types de courbes paramétriques. On obtient ainsi les *courbes de Bézier* [GR74, Bé86, FPDM14] où les ϕ_i sont des polynômes de Bernstein, les *B-splines* [dB78], ou encore les *NURBS*, pour *Non Uniform Rational B-Splines* [PT97]

Les surfaces paramétriques, quant à elle, se définissent par une équation paramétriques à deux paramètres, de type

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \phi_i(u) * \psi_j(v) * P_{i,j},$$

avec u et v les paramètres, ϕ_i et ψ_j les fonctions, et $P_{i,j}$ les pôles. Une surface paramétrique est ainsi la paramétrisation de la déformation d'une courbe.

L'intérêt des courbes et surface paramétriques est de permettre la création de formes quelconques (dans la limite imposée par les points de contrôle et le maillage), comme les aubes de turbines présentées à la figure 2.5, ainsi que la bonne maîtrise des formes engendrées grâce aux points de contrôles, qui permettent aussi une modification des courbes et surfaces. Cependant, à cause de la complexité des équations, certains traitements, comme par exemple le calcul du positionnement d'un point par rapport à une surface, peuvent être complexes.

2.2.3 Modèles topologiques

Les modèles topologiques permettent de représenter des objets subdivisés en *cellules* de différentes dimensions, comme les volumes, les faces, les arêtes, les sommets, ou d'autres

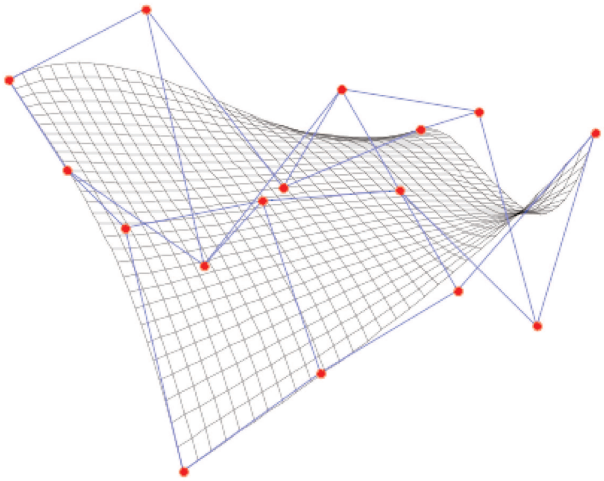
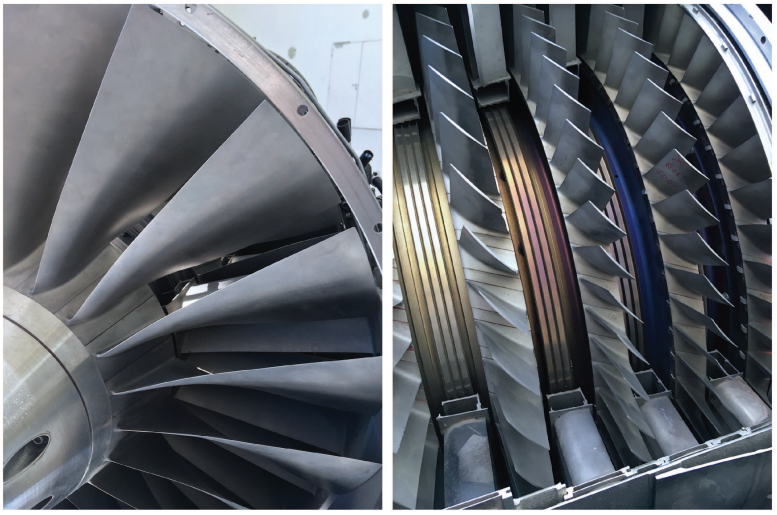


Figure 2.4 : Surface de Bézier



(a) **(b)**

Figure 2.5 : Aubes de turbine d'avion, Hall de l'ENSMA, Chasseneuil du Poitou

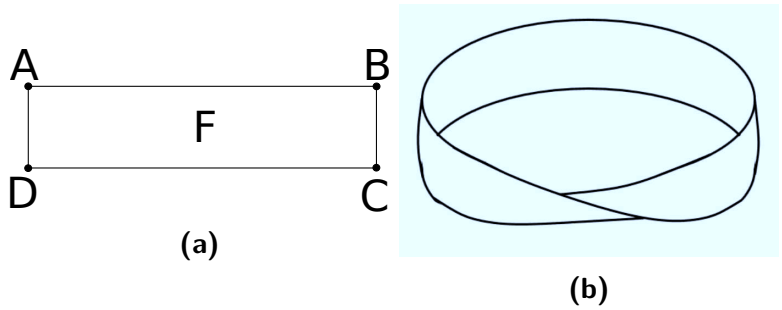


Figure 2.6 : Ruban de Möbius. Pour réaliser le ruban de Möbius, on utilise un patron rectangulaire (a). En liant les sommets A et B et les sommets C et D, on obtiendrait un anneau. Mais en liant les sommets A et C et les sommets B et D, on obtient un ruban de Möbius (b)

types de cellules en fonction des modèles, (section 2.2.3). Ces cellules sont liées entre elles par des relations topologiques appelées *liaisons*. Grâce à ces subdivisions, ce modèle décompose la structure d'objets des éléments représentés. Il est à noter que les modèles BRep évoqués à la section 2.2.1.2 sont définis par les subdivisions de leurs bords. Là où ces derniers représentent un assemblage de faces séparant des volumes, les modèles topologiques représentent un assemblage de volumes et les relations topologiques entre les différentes cellules qui composent ces modèles.

Afin d'obtenir une représentation géométrique d'un modèle topologique, il est nécessaire d'y associer des plongements, en définissant, par exemple, un plongement coordonnées 3D pour chaque sommet ou la couleur des éléments... Sans ces plongements, il ne peut y avoir de représentation géométrique.

Ce modèle permet de représenter des objets à la fois orientables ou non orientables, comme le ruban de Möbius en 2D (figure 2.6) ou une bouteille de Klein en 3D. La particularité d'une surface non orientable, comme par exemple le ruban de Möbius est que le parcours de sa surface permet de passer de l'intérieur à l'extérieur. C'est ce qu'on appelle une surface *non orientable*. À l'inverse, sur l'anneau, les deux côtés (intérieur et extérieur) sont bien distincts, c'est ce qu'on appelle une surface *orientable*. En 3 dimensions, on peut citer le tore, orientable, et la bouteille de Klein, non orientable.

On peut classer les modèles topologiques selon deux critères [LFB07] :

— le type de cellules
  — cellules *régulières*, comme les simplexes (triangles, tétraèdres ...), les simplexes cubiques (carrés, cubes ...), les simploïdes, ou les complexes simpliciaux abs-

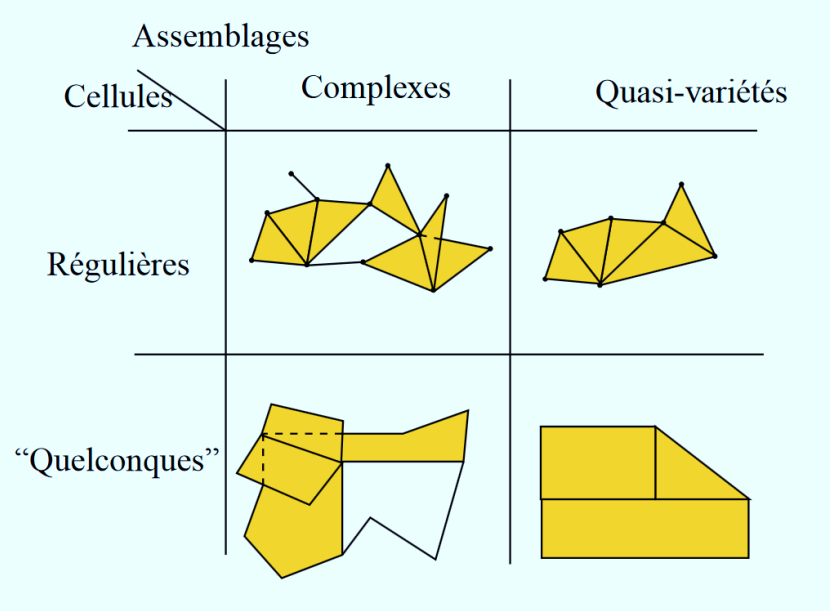


Figure 2.7 : Catégories de modèles topologiques, extrait de [LFB07]

traits [CDFM+94] qui sont des produits cartésiens de simplexes...

— cellules *quelconques*, permettant une plus grande diversité que les précédentes, mais vérifiant néanmoins les propriétés topologiques assurées par la définition du modèle. On peut citer dans cette catégorie les modèles ordonnés, comme les cartes généralisées [Lie94] ;

— le type d'assemblage. On peut distinguer les modèles permettant la multi-incidence ou non des cellules. Par exemple, une boucle, qui est une arête dont les sommets des bords sont confondus. Dans les deux cas, on peut distinguer deux sous-catégories :

— les *quasi-variétés*, ou *quasi-manifolds*. On peut définir une quasi-variété cellulaire de dimension n comme un ensemble de cellules de dimensions n, liées entre elles par des cellules de dimension $n-1$. Cette cellule de dimension $n-1$ ne peut elle-même être incidente qu'à deux cellules de dimension n. Les cartes généralisées font partie de cette catégorie.

— les assemblages *complexes*, ou *non-manifolds*. On peut citer comme exemples les chaînes de cartes [EL93] et les ensembles semi-simpliciaux [LP16].

La figure 2.7 montre des exemples de modèles topologiques, un par combinaison de catégories.

Cartes généralisées

Le modèle des *cartes généralisées*, ou *G-Cartes*, est un modèle topologique qui, comme nous l'avons vu dans la section précédente, se classe dans la catégorie des modèle à cellules quelconques et à assemblage de quasi-variétés. Une G-Carte est un graphe, dont les nœuds sont appelés des *brins*, et dont les arcs sont appelés des *liaisons alpha* (ou plus simplement *alpha*). Le modèle des cartes combinatoires est décrit pour la première fois par Vince en 1983 [Vin83].

Les G-cartes peuvent être vues comme l'éclatement d'un objet le long de ses bords. Ainsi, la figure 2.8 nous présente deux faces en 2 dimensions. L'image 2.8a présente le modèle en représentation géométrique. En éclatant le bord entre les faces, on obtient la figure 2.8b, avec les faces liées par des liaisons α_2, puis une seconde fois on éclate le bord entre les arêtes (figure 2.8c, avec les liaisons α_1 entre les arêtes), et enfin, on éclate le "bord" entre les sommets pour obtenir les brins, le plus petit composant d'une G-Carte (figure 2.8d avec les liaisons α_0 entre les brins).

Definition 1 (*Carte généralisée*)

Soit n un entier positif. Une n-G-Carte est définie comme un (n+2)-uplet $G = (B, \alpha_0, ..., \alpha_n)$, où :

- *B est un ensemble fini de brins*
- *$\alpha_0, ..., \alpha_i, ..., \alpha_n$ sont des permutations sur B, définies par :*
  - *Pour $0 \leq i \leq n$, α_i est une involution ; si B est vide, α_i n'est pas définie.*
  - *Pour $0 \leq i \leq i+2 \leq j \leq n$, $\alpha_i \alpha_j$ est une involution.*

Les G-Cartes sont ainsi définies de la même manière, quelle que soit la dimension du modèle.

Definition 2 (*Type d'orbite*)

Soit $G = (B, \alpha_0, ..., \alpha_n)$ une nG-Carte, et ϕ un ensemble de m permutations définies sur B. On appelle ϕ un type d'orbite.

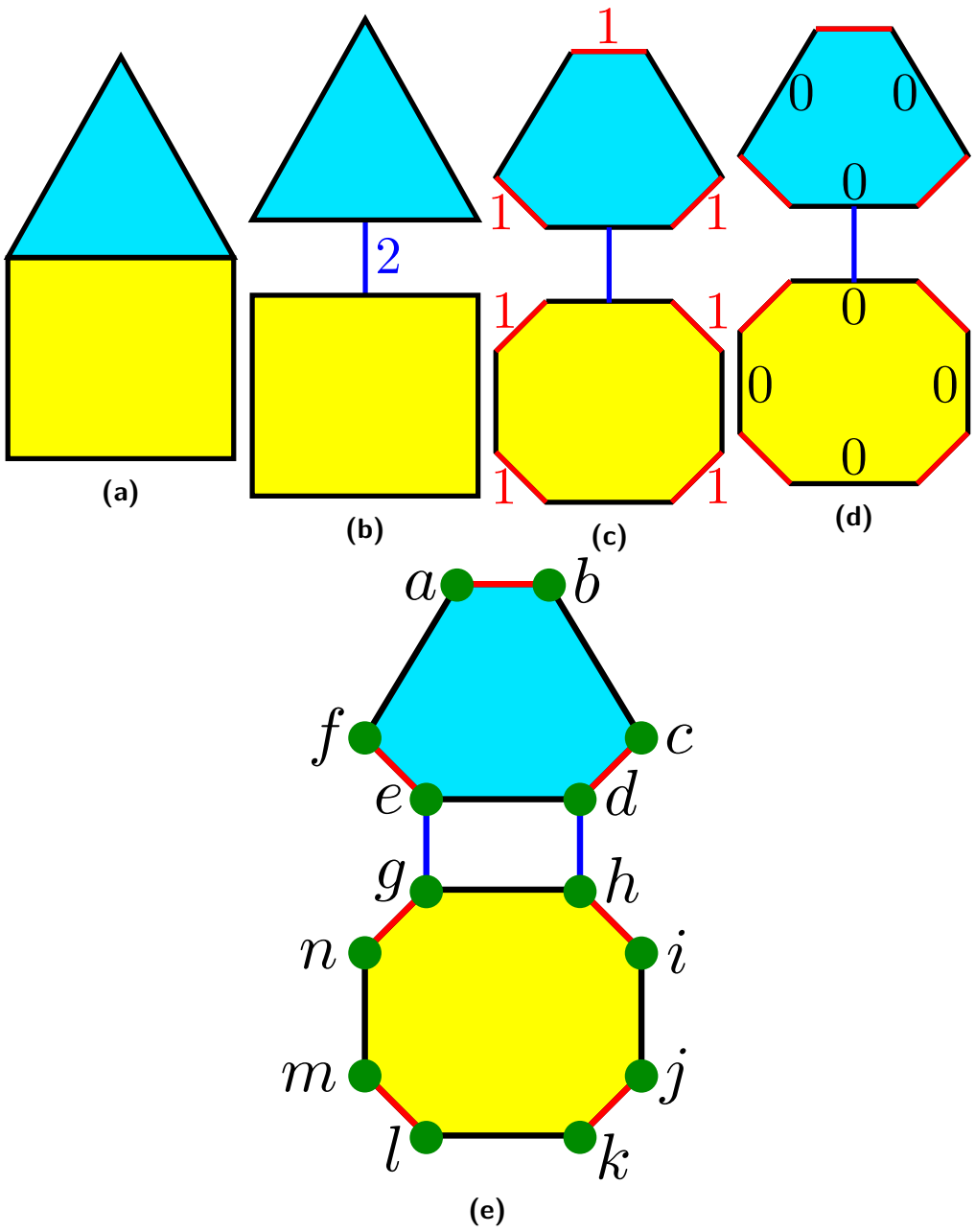


Figure 2.8 : Éclatement d'un modèle en dimensions successives

Definition 3 (*Orbite*)

Soit $G = (B, \alpha_0, ..., \alpha_n)$ une nG-Carte, b un brin quelconque de B, et ϕ un ensemble de m permutations définies sur B. On appelle orbite de b par rapport à ϕ l'ensemble, noté $b.\phi$ et défini comme :

$$b.\phi = \{b' \in B | 1 \leq k \leq m, \exists \phi_1 \in \phi, ..., \exists \phi_k \in \phi, b' = \phi_k(...(\phi_1(b)))\}$$

L'orbite $b.\phi$ correspond donc à l'ensemble des brins de B que l'on peut atteindre à partir de b par une composition de permutations de ϕ et de leurs inverses.

On note $\langle x, y, z \rangle$ le type d'orbite correspondant aux permutations sur α_x, α_y et α_z.

La figure 2.8e nous permet de distinguer chaque brin composant notre G-Carte, ainsi que ses orbites. Par exemple, la face $g.\langle 0, 1 \rangle$, comprend les brins $\{g, h, i, j, k, l, m, n\}$, tous atteignables *via* les permutations de liaisons α_0 et α_1 au départ de b. De même, on peut définir l'arête liée à g, $g.\langle 0, 2 \rangle = \{g, h, d, e\}$, le sommet lié à g, $g.\langle 1, 2 \rangle = \{g, n, e, f\}$. D'autres orbites existent, moins conventionnelles que les cellules, comme le sommet restreint à une face lié à g, $g.\langle 1 \rangle = \{g, n\}$ ou l'arête restreinte (ou demi-arête) à la face liée à g, $g.\langle 0 \rangle = \{g, h\}$.

Des travaux ont été menés pour définir une structure hiérarchique basée sur les G-Cartes, les *G-Cartes pyramidales* [GSDL06]. Celles-ci permettent de grouper plusieurs G-Cartes dans une représentation multi-échelle, ce qui permet de voir plusieurs variations du modèle, et, par exemple, de représenter différents niveaux de détails (afin de permettre un affichage plus léger quand on est plus loin de l'objet en stockant moins de détails), ou l'évolution d'un modèle à travers le temps en utilisant la quatrième dimension (3D + temps).

Pour nos travaux, nous avons fait le choix du modèle des G-Cartes. En effet, comme nous le verrons dans la section 2.5, ce modèle est celui utilisé par le modeleur que nous voulons utiliser, *Jerboa*. De plus, ce modèle utilise une manière homogène de nommer les orbites en dimensions, ce qui nous sera utile pour le système de nommage persistant que nous avons élaboré, et qui sera abordé dans la section 3.1.

2.3 Modèles paramétriques

Afin de pouvoir faire varier un modèle, il est nécessaire que ses paramètres puissent être modifiés. C'est ainsi qu'est apparue l'idée de *modèle paramétrique*. Le fait de créer ces variations est appelé *réévaluation*, ou *rejeu*. On appelle jeu initial la première itération du

modèle paramétrique, avant variations.

Un modèle paramétrique est conçu en deux couches :

— La *couche géométrique*, qui correspond à la représentation du modèle, et utilise typiquement l'un des modèles évoqués dans la section 2.2.

— La couche de la *spécification paramétrique*, qui correspond à l'historique de construction qui contient toutes les opérations permettant de créer le modèle, ainsi que leurs paramètres.

Grâce à la spécification paramétrique, on peut *réévaluer* le modèle, c'est-à-dire le générer à nouveau en changeant certains de ses paramètres, pour en créer des variations. Dans le domaine de la CAO, ces modèles paramétriques sont utilisés depuis des années afin de réaliser des réévaluation automatiques [Rol91, AM95, Kri95, BS01, HK01].

On peut distinguer deux catégories de modèles paramétriques, l'approche équationnelle et l'approche fonctionnelle.

2.3.1 Approche équationnelle

Dans cette approche, un solveur résoud des contraintes (parallélisme, tangence, perpendicularité, distance,...) indiquées par l'utilisateur. Ces contraintes font le lien entre les différentes entités du modèle. Les entités sont les éléments géométriques du modèle (sommets, arêtes...). Ce modèle est réalisé grâce à une esquisse, sur laquelle s'appliquent les contraintes. Toutes les équations sont ensuite résolues en utilisant soit des méthodes géométriques, comme avec Geom3 [But79], Progé [Sch93] (décrit comme un logiciel permettant une résolution "à la règle et au compas") ou encore [DMS98], [MF09] ou [Mou16], soit des méthodes combinatoires [Owe91] ou [YMJMSMC14], soit les méthodes numériques probabilistes [FMJ05].

En faisant varier les valeurs des contraintes (quand celles-ci en ont), on peut ainsi modifier le modèle, par exemple celui de la figure 2.9 qui présente deux versions d'un même modèle, avec une variation d'une de leurs contraintes concernant la longueur de l'arête supérieure. De plus, les contraintes étant résolues après que l'esquisse soit réalisée, l'utilisateur peut faire une esquisse grossière et la contraindre ensuite afin d'obtenir le résultat voulu.

Le principal avantage de cette approche est sa simplicité de mise en œuvre pour la réalisation d'esquisse. Cependant, la résolution des équations peut être plus ou moins

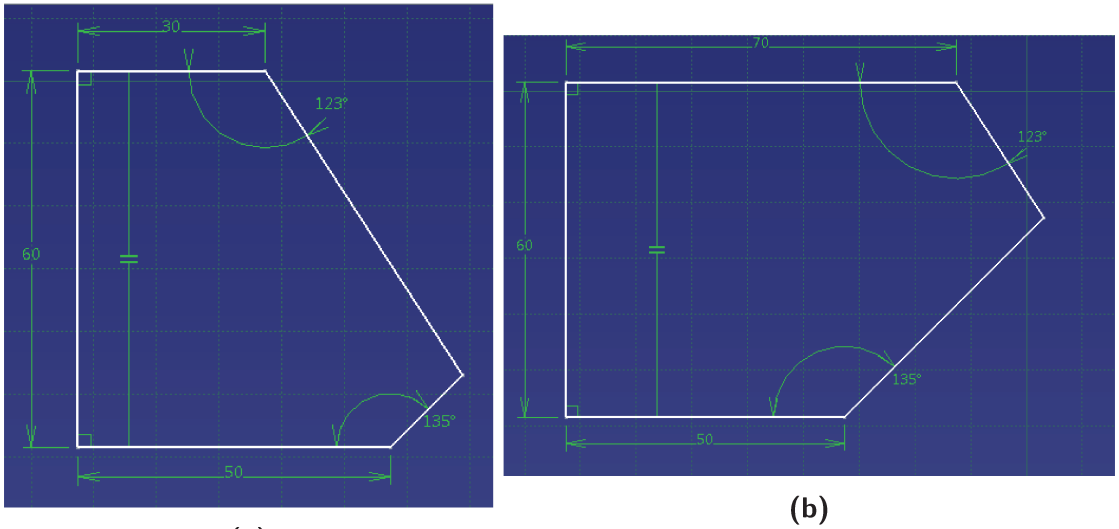


Figure 2.9 : Changement de la valeur d'une contrainte et impact sur un modèle

rapide en fonction de la puissance de l'ordinateur sur lequel le modèle est réalisé, surtout au passage en 3D, qui augmente le nombre de contraintes, et donc d'équations. De plus, comme tout système d'équations, s'il n'y a pas autant d'inconnues que d'équations, il peut y avoir des problèmes pour la résolution de ces équations (système sous-contraint ou sur-contraint).

2.3.2 Approche fonctionnelle

Dans cette approche, on utilise plusieurs fonctions de modélisation successives. Ces fonctions permettent de réaliser des variations du modèle grâce à la modification de leurs paramètres.

On peut ainsi créer une droite $D = Droite(P_1, P_2)$ avec P_1 et P_2 deux points de passage de la droite, définis à une étape précédente, ou un pavé $P = Pavé(L, l, P)$ avec L sa longueur, l sa largeur et P sa profondeur. P_1, P_2, l, l et P sont des paramètres qu'il suffit de changer pour réévaluer le modèle. La figure 2.10 présente la couche géométrique associée à la spécification paramétrique suivante :

— $1 - Pavé(5, 5, 5)$
— $2 - Rainure(f, 2, 6)$, avec f la face d'application, 2 la largeur de la rainure, et 6 sa longueur

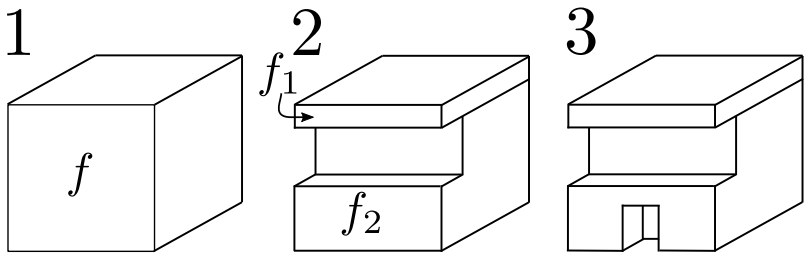


Figure 2.10 : Exemple de modélisation paramétrique en 3 étapes.

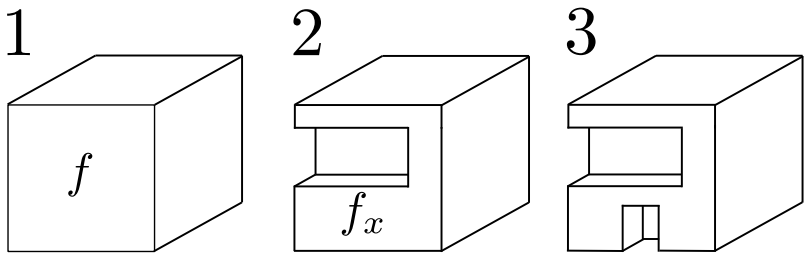


Figure 2.11 : Réévaluation de la spécification paramétrique de la figure 2.10.

— $3 - Rainure(f_2, 1, 1)$

Par rapport à l'approche équationnelle, le nombre de paramètres augmente moins vite que le nombre de contraintes, de manière générale.

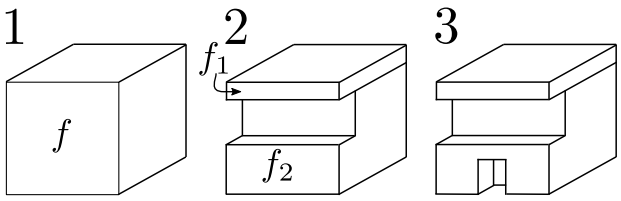
Cependant, chaque fonction utilise comme paramètres des entités du modèle, définis à une étape précédente, comme f et f_2 dans notre exemple. En réévaluant le modèle, ces entités peuvent ne plus exister. Ainsi, en réévaluant notre spécification en modifiant la valeur de la longueur de la première rainure, de 6 à 4 , nous obtenons le modèle de la figure 2.11, à partir de la spécification paramétrique suivante :

— $1 - Pavé(5, 5, 5)$

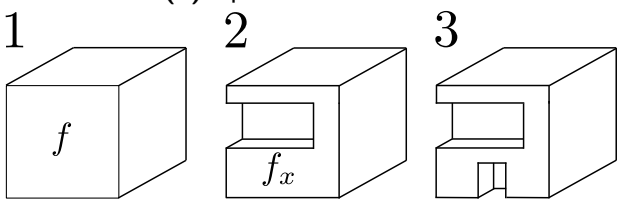
— $2 - Rainure(f, 2, \mathbf{4})$

— $3 - Rainure(f_2, 1, 1)$

Or, dans la spécification initiale, l'opération 3 prenait en paramètre d'entrée f_2, qui n'existe plus dans la réévaluation, étant donné que la rainure insérée par l'opération 2-Rainure était plus courte, modifiant simplement f en f_x. Il faut donc un système d'identification robuste et persistant pour que le logiciel reconnaisse que f_x et f_2 se correspondent. C'est ce que nous verrons dans la section 2.4.



(a) Spécification Initiale



(b) Spécification Réévaluée

Figure 2.12 : Exemple d'utilité du nommage pour la réévaluation

2.4 Problème de la nomination persistante

Afin de réaliser le rejeu d'un modèle paramétrique, il faut pouvoir apparier les entités. Dans cette section, nous verrons d'abord en quoi le nommage persistant permet de suivre et apparier les entités, puis les concepts communs aux différentes méthodes de nommage existantes, et enfin, certains types de nommage existant.

2.4.1 Nécessité du nom et de l'appariement

L'utilisation d'un modèle paramétrique entraîne l'utilisation d'un système permettant d'apparier les différents entités au jeu initial et au rejeu. Ce système est appelé *nommage persistant*. En effet, l'utilisation d'un identifiant incrémenté à la création de l'entité (ID) seul ne suffit pas, car comme nous l'avons vu dans l'exemple 2.11, les changements de paramètres peuvent conduire à la création de plus ou moins d'entités, et donc des identifiants différents entre le modèle initial et le modèle rejoué. Ici, par exemple, f_2 aurait l'ID F_a, et f_x l'ID F_b.

Le nommage persistant est alors nécessaire afin de mettre au point un système d'identification robuste des entités en dehors du modèle paramétrique, une troisième couche qui vient s'ajouter aux deux déja existantes, présentées dans la section 2.3 [BA10], [MP02].

La figure 2.12 est un rappel du modèle étudié comme exemple dans la section précédente.

Comme nous l'y avons montré, la face f_2 n'existe pas après réévaluation, on ne sait donc pas sur quelle face appliquer la deuxième insertion de rainure.

Afin de pallier à cela, le nommage persistant s'intercale comme une *troisième couche* de la modélisation paramétrique, entre la couche géométrique et couche de la spécification paramétrique. Il permet de donner un *nom* aux entités, nom qui ne sera pas affecté par les changements appliqués au modèle, d'où le terme persistant.

Un nommage persistant permettrait, ici, de constater que f_2 est issue de f (par scission) et que, f_x étant également issue de f (par modification), les deux pourraient être appariées. La deuxième insertion de rainure pourrait ainsi être effectuée sur f_x.

2.4.2 Solutions proposées

Le concept de nom persistant a été introduit pour la première fois en 1994 par Capoyleas et al. [CCH96] et en 1995 par Kripac [Kri95]. Ils y posent certaines bases du nommage persistant, comme la non-utilisation des IDs car non stables, mais également, la non-utilisation des pointeurs (des variables contenant une adresse mémoire, qui stocke les informations concernant une entité, ici, l'entité à nommer), qui changent d'une itération à une autre. Ils soulignent également un point important : la relation de noms persistants n'est pas une relation 1-1. En effet, dans le cas où une entité serait scindée, elle pourrait correspondre à plusieurs entités au rejeu, et dans le cas d'une fusion, plusieurs entités correspondraient, au rejeu, à une seule.

2.4.2.1 Concepts communs

Dans l'étude [MP02], les auteurs présentent un certain nombre de "concepts communs" que l'on retrouve dans la plupart des approches traitant du problème de nommage persistant.

Entités invariantes et contingentes Le premier de ces concepts est la différence entre entités *invariantes* et entités *contingentes*. Celles-ci sont définies comme suit par Agbodan et al. dans [AMP99]

Definition 4 (*Entité invariante*)

Une entité invariante est une entité géométrique ou topologique qui peut être complètement et sans ambiguïté caractérisée par la structure d'une opération de modélisation et ses paramètres d'entrée, indépendamment des valeurs impliquées.

Definition 5 (*Entité contingente*)

Une entité contingente est une entité topologique ou géométrique résultant de l'interaction entre le modèle géométrique préexistant et les entités invariantes issues de la dernière opération de modélisation.

Par exemple, sur la figure 2.12, le cube de l'étape 1 est composé d'entités invariantes, tandis que les entités résultant de la création de la rainure à l'étape 2 sont des entités contingentes.

Comme vu précédemment, entre la spécification initiale et la spécification réévaluée, on ne peut pas prévoir quelles entités contingentes sont affectées (il y a une face contingente de plus à la réévaluation par exemple), alors que, quelles que soient les valeurs des paramètres de la spécification paramétrique, les mêmes entités invariantes seront systématiquement créées au jeu et au rejeu (par exemple le cube sera toujours constitué des mêmes entités topologiques).

Nomination des entités invariantes

Le fait d'avoir deux types d'entités entraîne également généralement deux types de nommage, un pour les entités invariantes et un pour les entités contingentes.

Pour les entités invariantes, le nommage est lié aux structures topologiques des opérations de modélisation qui les affectent (trajectoire d'une extrusion par exemple). Pour cela, il faut pouvoir identifier ces structures de manière robuste, afin que le nommage ne soit pas sujet à des variations.

Par exemple, Capoyleas et al. [CCH96] propose de nommer les entités résultantes d'une extrusion en plusieurs étapes (figure 2.13). Tout d'abord, il identifie les sommets de l'esquisse (ici, $v_1...v4$), ses arêtes ($e_1...e_4$) et sa face f. Il nomme ensuite les arêtes (resp. les faces) créées par l'extrusion à partir des sommets (resp. des arêtes) de l'esquisse (ex : $e(v_1)$, resp. $f(e_1)$). Enfin, pour lever l'ambiguïté sur les sommets $v_1...v4$, les arêtes $e_1...e_4$ et la

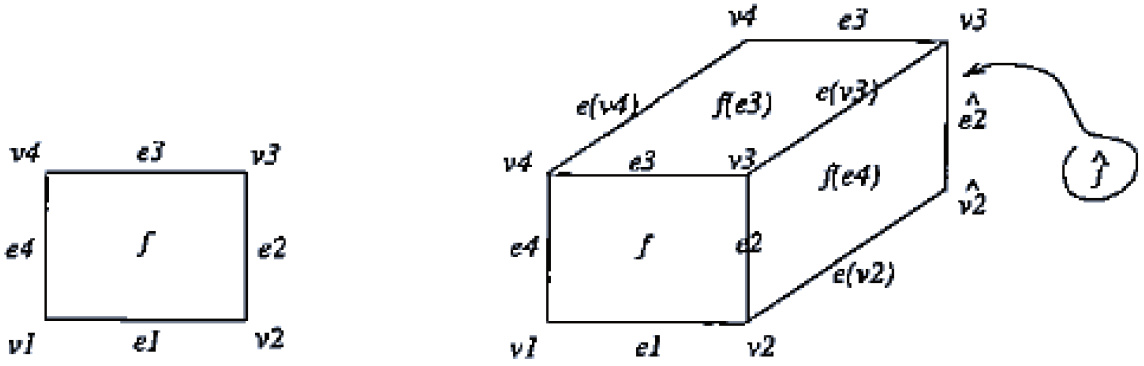


Figure 2.13 : Nommage des entités invariantes pour une extrusion, extrait de [CCH96]. A gauche : l'esquisse. A droite : le résultat de l'extrusion de cette esquisse.

face f, qui apparaissent maintenant deux fois, il utilise la direction de l'extrusion, la moitié de ces entités se trouvant à l'origine de l'extrusion, et l'autre à sa fin.

Nomination des entités contingentes

L'imprévisibilité des entités contingentes les rend plus difficiles à nommer. Pour pallier à ce problème, il faut les nommer par rapport à une entité plus stable. Le nommage des entités contingentes est donc généralement basé sur les noms des entités invariantes. On pourrait également envisager de les nommer par rapport à leur voisinage, mais celui-ci n'est pas forcément composé d'entités invariantes, on ne ferait donc que repousser le problème.

Capoyleas et al., toujours dans [CCH96], propose de considérer les opérations comme des *proto-opérations*, et de nommer de manière invariante les entités créées par l'opération avant son application. Sur la figure 2.12, la rainure serait ainsi le résultat de l'interaction entre le cube et un pavé représentant la rainure. Ce pavé est composé d'entités invariantes, qui peuvent être nommées. Les nouvelles entités créées par cette interaction récupéreraient donc ces noms. En cas d'ambiguïté (sommets, arêtes ou faces qui se chevauchent), il propose de nommer l'entité en fonction de la géométrie existante.

Historique dans la nomination

Le quatrième concept commun est la notion d'historique. Il permet de suivre l'évolution

des entités. En effet, le nommage persistant reposant sur l'idée d'ancêtres invariants, un historique d'évolution est nécessaire pour pouvoir apparier deux entités *via* leurs ancêtres. En effet, celui-ci est nécessaire pour l'appariement. Comme nous l'avons vu dans l'exemple de la figure 2.12, le fait de savoir que f_2 et f_x sont issues de la même face ancêtre f a permis de réaliser le rejeu.

Afin de réaliser cet historique, il existe deux solutions. Tout d'abord, l'inscrire directement dans le nom persistant des entités, par exemple en concaténant des noms successifs issus de chaque modification. Chaque entité porte ainsi l'histoire de son évolution et la propage à ses descendants [CCH96], [WZZ01]. Dans notre exemple, f_x pourrait par exemple être nommé $c(f) : m(f)$, indiquant tout d'abord la création de f puis sa modification, et f_2 $c(f) : s(f)$, indiquant la création de f puis sa scission.

La seconde solution est de créer un arbre d'historique [Kri95]. La figure 2.14 présente l'arbre d'historique de Kripac. Cet arbre est créé au jeu initial. Quand une opération est réévaluée, on vérifie si les noms de ses paramètres apparaissent dans le graphe. Si c'est le cas, on réévaluera l'opération sur tous les enfants de la branche qui a ce nom pour origine. Ainsi, si une opération est réalisée sur 13.2 au jeu initial, celui-ci ayant pour enfant 14.1 au rejeu, c'est sur 14.1 qu'on réévaluera l'opération.

Architecture à 3 couches Le dernier concept commun, dont nous avons déjà parlé dans l'introduction de cette section, est l'architecture à 3 couches d'un système paramétrique intégrant un système de nommage persistant :

- La couche géométrique, contenant la représentation géométrique de l'objet modélisé.
- La couche de la spécification paramétrique, contenant l'historique des opérations de modélisation utilisées pour construire le modèle ainsi que leurs paramètres.
- La couche de nommage, permettant d'identifier et d'apparier chaque entité.

2.4.2.2 Approches existantes

Dans cette partie, nous abordons des travaux précurseurs ou ayant apporté des nouveautés au nommage. Certains, comme [CCH96] et [Kri95] ont déjà étés abordés dans la section précédente, mais nous les détaillons ici.

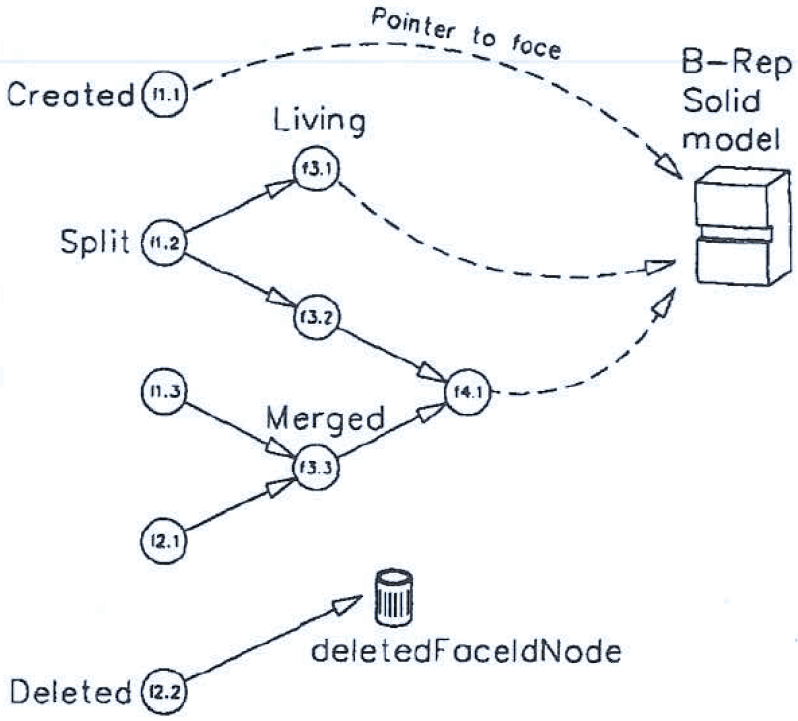


Figure 2.14 : Graphe de suivi des entités chez Kripac, extrait de [Kri95]. Les faces $f1.1$, $f1.2$, $f1.3$, $f2.1$ et $f2.2$ sont d'abord créées, puis $f1.3$ et $f2.1$ sont fusionnées en $f3.3$, $f1.2$ scindée en $f3.1$ et $f3.2$ et $f2.2$ supprimée. Enfin, $f3.2$ et $f3.3$ sont fusionnées en $f4.1$. A la suite de ces étapes, il reste $f1.1$ $f3.1$ et $f4.1$ dans le modèle.

Capoyleas et al., 1994

Ces travaux, réalisés en 1994 [CCH96] et complétés par la thèse de Chen en 1995 [Che95], font partie des premiers concernant le nommage persistant.

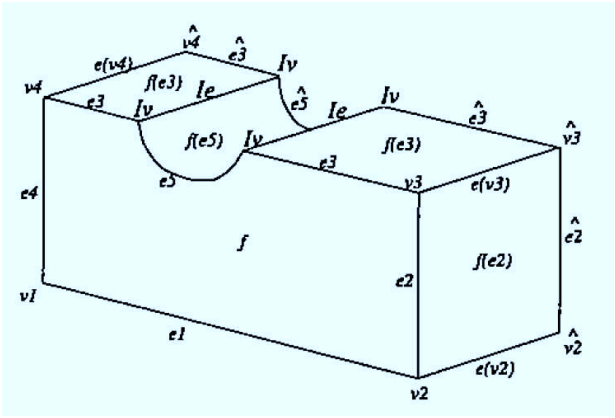
En plus des extrusions, pour lesquelles nous avons vu son mécanisme de nommage des entités invariantes dans la section précédente, il propose un système de nommage pour les entités issues d'une révolution, en utilisant le profil, l'axe, et la direction utilisés pour la réaliser.

Il propose également un système permettant de nommer les entités en fonction des opérations qui les ont modifiées, comme nous l'avons vu dans la section 2.4.2.1, avec, par exemple, le nommage du résultat d'une rainure via 1) les entités existantes dans le modèle 2) le pavé que l'opération de rainure enlève au modèle. Il appelle cela la fusion d'entités nommées.

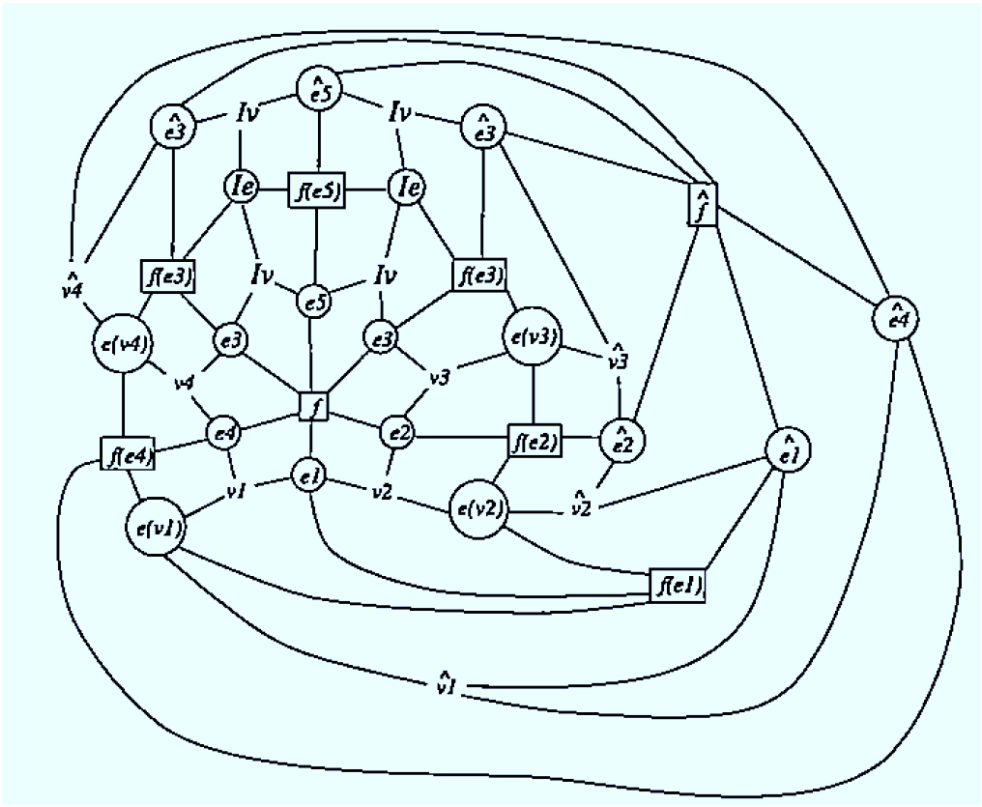
Chen ajoute à cela la notion discrimination des entités, c'est à dire, de lever une ambiguïté sur leur nom persistant. Il propose par exemple l'utilisation d'un demi espace, solution rejetée à cause de la non-existence réelle de ces demis espaces, qui peut les rendre difficile à appréhender pour l'utilisateur, ou la discrimination des faces par rapport à leur "position" dans l'opération (face-départ et face-arrivée pour une extrusion, par exemple) [CH94].

Afin de caractériser les entités, ils déterminent également la notion de *contexte topologique*. Ils réalisent un *graphe contextuel* permettant de lier tous les éléments incidents entre eux. Un exemple de ce graphe est présenté sur la figure 2.15. Ils déterminent ensuite le *contexte étendu* souhaité, 1 pour le voisinage immédiat, 2 pour le voisinage immédiat et son voisinage... Afin de discerner deux entités, ils stockent ainsi des listes des noms des voisins, sous la forme suivante : $[nomEntité], [nomsEntitésContexte1], [nomsEntitésContexte2]$... On réduit ensuite ce nom en enlevant les noms non discriminants, tout en gardant au moins un nom dans chaque liste. Ainsi, les deux arêtes I_e de la figure 2.15a sont discriminées par leur deuxième voisinage. On a : $I_e : [0, I_e, 1], [1, f(e_3), 1], [2, e(v_4), 1]$ pour la première, et $I_e : [0, I_e, 1], [1, f(e_3), 1], [2, e(v_3), 1]$ pour la deuxième, les chiffres entourant le nom permettant de donner le degrés de voisinage de l'entité pour le premier, et le nombre de nœud dans la classe pour le deuxième.

Ce système basé sur les faces, n'est pas généralisable en dimension (un modèle de dimen-



(a)



(b)

Figure 2.15 : Graphe de contexte, issus de [CCH96]. a : modèle, b : graphe de contexte lié à a.

sion 4 aurait un nommage basé sur les volumes), ni homogène, car le nommage n'est pas le même pour une face, une droite ou un sommet. De plus, le fait que la discrimination repose sur les voisinages à un certain éloignement peut faire craindre de ne pas pouvoir retrouver une entité si le contexte change trop. Enfin, la notion d'appariement entre entités au jeu initial et entités au rejeu n'est pas abordée.

Cependant, on peut noter une première forme d'historique dans ce système, les entités issues d'une extrusion étant, comme nous l'avons vu dans la section 2.4.2.1, nommées par rapport à l'entité qu'elles extrudent. L'arête issue d'un sommet v_1 est par exemple nommée $e(v_1)$.

Kripac, 1995

Le système développé par Kripac nomme différemment les éléments sommets, arêtes, et faces. Il utilise les numéros d'étapes des opérations pour nommer les éléments. Il s'appuie sur les faces, qu'il décrit comme "les bords des volumes qui composent un modèle", car les arêtes et sommets peuvent ensuite être vues comme des intersections de faces. Il détermine dans un premier temps le nom persistant des faces, basé entre autres sur les numéros d'étapes des opérations de la spécification. Il utilise ensuite ces noms de faces pour nommer les arêtes et les sommets, *via* les noms des faces qui leurs sont incidentes.

Il nomme ainsi les faces :

$$FaceId(f) = [stepID, faceIndex, surfaceType]$$

avec $stepID$ l'ID de l'opération qui crée f, $faceIndex$ l'ID de f à sa création, $surfaceType$ le type de surface de f.

Les arêtes sont ensuite nommées :

$$EdgeID(e) = [adjFaceIds, endFaceIds_{0,1}, edgeIntersCode]$$

avec $adjFaceIDds$ les IDs des faces incidentes à e, dans l'ordre cyclique, $endFaceIds_{0,1}$ les IDs des faces incidentes à e à ses extrémités, $edgeIntersCode$ le code d'intersection de l'arête. Ce code est basé sur la position relative des faces incidentes à cette arête, ce qui le rend non sensible à des opérations comme un déplacement ou un changement d'échelle, ce qui ne serait pas le cas s'il était basé sur la position en coordonnées absolues.

Et enfin, les sommets sont nommés :

$$VertexId(v) = [adjFaceIds, vertexIntersCode]$$

avec $adjFaceIds$ les IDs des faces incidentes à v, $vertexIntersCode$ le code d'intersection du sommet, à nouveau basée sur la position relative à celui-ci des faces qui lui sont incidentes.

Il suit l'évolution (scission, modification, fusion…) des faces, bases de son nommage, *via* un graphe, présenté sur la figure 2.14. Enfin, l'appariement est réalisé, pour les faces, en associant celles ayant le même ID au jeu initial et les enfants de celles ayant cet ID au rejeu. Pour les arêtes (resp. sommets), on les associe à celles (resp. ceux) ayant le même ID au jeu initial, ou, si aucune arête (resp. sommet) ne satisfait cette condition, aux arêtes (resp. sommet) du jeu initial ayant au moins 2 faces incidentes commune avec celle (resp. celui) du rejeu. Dans ce dernier cas, il faut aussi que les $edgeIntersCode$ (resp. $vertexIntersCode$) coincident.

Kripac propose à cet effet un système permettant d'indiquer le degré d'appariement souhaité, c'est-à-dire, le nombre minimum de noms persistants de faces en commun nécessaires.

Cet article est l'un des premiers à présenter le nommage et d'en poser toutes les bases (robuste, impliquant un appariement, fonctionnant pour tout type d'entité), mais présente également certaines limitations, comme un nommage différent en fonction du type d'entité. De plus, il n'est *a priori* pas généralisable en dimension, est basé sur les voisinages (les arêtes et sommets sont caractérisés par leurs faces incidentes) : si le voisinage est modifié suite au rejeu, on peut donc craindre que les opérations ne soient plus applicables.

Wu et al., 2000

Les travaux de Wu et al. [WZZ01] se basent sur ceux de Kripac et Capoyleas évoqués dans les paragraphes précédents, et les complètent. Ils permettent d'utiliser les opérations booléenne, ce qui complète les travaux de Capoyleas, et prennent en compte les ambiguïtés, ce qui complète Kripac. Ces travaux se retrouvent donc à cheval sur ces deux approches.

Wu reprend l'idée de garder une forme d'historique dans le nom, afin de connaître les origines d'un élément et donc de pouvoir l'apparier par la suite.

Le nommage de Wu est, comme celui de Kripac, basé sur les faces, en caractérisant les entités par leurs faces incidentes. Il nomme les faces issues d'opérations sur un profil 2D (face invariantes), comme les extrusions de la façon suivante :

$$ON(F) = [FeatID, FeatID_p, ID_{element}, FeatID_{Path}, ID_{trajectory}]$$

avec $FeatID$ l'ID de l'opération, $FeatID_p$ l'ID du profil, $FeatID_{Path}$ l'ID du chemin d'extrusion, $ID_{element}$, l'ID de l'arête du profil qui génère la face, et enfin $ID_{trajectory}$ l'ID du segment du chemin d'extrusion qui génère la face. Pour la face de départ et la face d'arrivée, qui ont le même nom, on les différencie par leur $ID_{element}$, -1 pour la face de départ et -2 pour celle d'arrivée.

Pour les faces issues de la modification d'autres entités (faces contingentes), comme par exemple un chanfrein issu de la modification d'une arête, il propose le nommage suivant :

$$ON(F) = [FeatID, FeatID_{FF1}, ID_{element1}, FeatID_{FF2}, ID_{element2}]$$

avec $FeatID$ l'ID de l'opération, $FeatID_{FF1}$ et $FeatID_{FF2}$ les IDs des deux faces qui étaient incidentes à l'arête éclatée, $ID_{element1}$ et $ID_{element2}$ les $ID_{element}$ de F.

Enfin, les faces basées sur des caractéristiques, comme par exemple des faces issues d'une répétition linéaire (c'est à dire, répétées à un intervalle régulier le long d'une droite), sont nommées :

$$ON(F) = [FeatID, FeatID_p, ID_{element}, FeatID_{path}, FeatID_{trajectory}, InsNum]$$

avec $InsNum$ le nombre séquentiel désignant l'instance à laquelle appartient F, et $FeatID_{trajectory}$ et $FeatID_{path}$ désignant le nombre d'instances de répétition dans chaque sens.

La figure 2.16 présente le résultat d'une répétition linéaire du cylindre de gauche. $F1$, $F2$ et $F3$ sont issues d'une extrusion, et nommées $ON(F1) = [6, 5, -2, 0, 0]$, $ON(F2) = [6, 5, -1, 0, 0]$ et $ON(F3) = [6, 5, ID_e, 0, 0]$. On obtient ainsi, pour les faces issues de la première et de la deuxième répétition, les noms $ON(F1)' = [7, 5, -1, 0, 1]$, $ON(F2)' = [7, 5, -2, 0, 1]$, $ON(F3)' = [7, 5, ID_e, 0, 1]$, et $ON(F1)'' = [7, 5, -2, 0, 2]$, $ON(F2)''[7, 5, -1, 0, 2]$, $ON(F3')' = [7, 5, ID_e, 0, 2]$.

Il propose également un système d'évolution des noms persistants, ce qui permet de propager l'historique :

— En cas de suppression d'une face, son nom est supprimé.
— En cas de modification d'une face sans scission ni fusion, elle garde son nom.
— En cas de scission, toutes les faces enfants récupèrent le nom de la face mère.
— En cas de fusion, la face enfant récupère le nom de la dernière de ses faces mères.

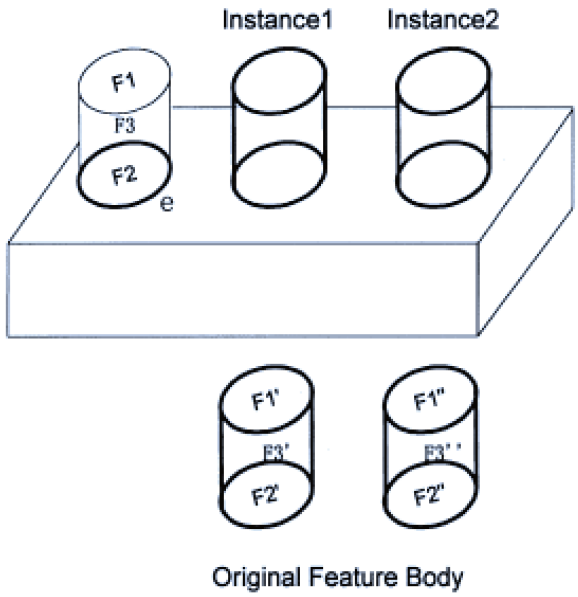


Figure 2.16 : Nommage chez Wu, extrait de [WZZ01]. Nommage des faces issues d'une répétition linéaire

On peut ainsi utiliser des opérations booléennes afin d'obtenir plus de modèles, l'insertion d'une rainure, par exemple, étant vue comme le résultat de l'opération booléenne qui enlève un pavé au modèle.

Enfin, afin de résoudre le problème d'ambiguïté sur le nom, Wu utilise la géométrie, et plus particulièrement, les coordonnées des sommets.

Le "nom réel" d'une face est ensuite obtenu en couplant son nom persistant à ses informations géométriques. Pour obtenir celui d'une arête, on utilise les noms réels de ses deux faces incidentes et ses informations géométriques. Enfin, le nom réel d'un sommet est obtenu en combinant le nom réel de ses arêtes incidentes et ses informations géométriques.

Comme indiqué dans l'introduction de cette partie, cette méthode fait le lien entre les méthodes de Kripac (opérations booléennes) et de Capoyleas (résolution des ambiguïtés, historique dans le nom).

Cependant, son nommage, basé sur les faces, n'est pas généralisable. De plus, sa résolution des ambiguïtés étant basée sur la géométrie, ce système peut parvenir à ses limites en cas de changement trop important de la géométrie.

Parmi des méthodes plus récentes, on peut citer celle de Farjana et al. [FMH15], [FHM16], basée sur celle de Wu, une entité y étant définie par une suite d'identifiants comme par

exemple celui du chemin d'extrusion ou d'une esquisse. Ainsi, on définit la création d'un cercle de 30 de rayon par $[esquisse, cercle1, 30]$. Les arêtes et sommets sont à nouveaux définis *via* leurs faces incidentes. Cette méthode a pu être implémentée [FHM16]. Cependant, comme d'autres, elle n'est ni homogène ni générale. De plus, l'implémentation ne semble pas présenter le rejeu, seulement le nommage. La manière dont le nommage permet de résoudre les ambiguïtés pouvant apparaître au rejeu n'est pas explicitée dans l'article, et si elle est à base géométrique comme pour le système de Wu (ce qui semble probable, des notions géométriques étant utilisées dans le nom), le système peut parvenir à ses limites en cas de trop grands changements géométriques.

Bidarra, 2005

Dans ses travaux, Bidarra [Bid05] veut présenter une alternative au nommage utilisant les bords des entités et leurs voisinages topologiques. En effet, il indique qu'en utilisant les bords, on finit avec une interdépendance des entités, pouvant entraîner des problèmes pour la précision du nommage. En effet, une ambiguïté sur une entité se propagera à toute celles qui l'utilisent pour leur nommage.

Le système de Bidarra se base sur des *caractéristiques*, ou *features*, définies comme : "des entités représentant les intentions de conception ou le sens de la géométrie d'une pièce ou d'un assemblage" [SM95], ou encore comme "des entités permettant de capturer explicitement les attributs d'ingénierie et les relations entre les entités définissant un produit" [MNS96].

Pour définir le nommage, il fait appel à trois catégories d'entités persistantes :

— Classes de référence : une référence est une entité annexe au modèle, nommée de manière unique, qui peut être associée à d'autres entités de la spécification par des contraintes géométriques. Les références fournissent des informations de position et d'orientation. On peut par exemple citer la classe de référence cylindrique ou la classe de référence planaire. Ces références permettent de créer et d'éditer des caractéristiques. Par exemple, pour créer une série linéaire de trous dans un modèle, on peut créer une référence planaire, puis créer les instances de trous dans le référentiel de ce plan. Pour déplacer tous les trous, il n'y aura alors plus qu'à déplacer le plan.

— Classes de caractéristiques déclaratives : une description structurée de toutes les propriétés d'une caractéristique donnée, qui définit un motif pour toutes ses instances.

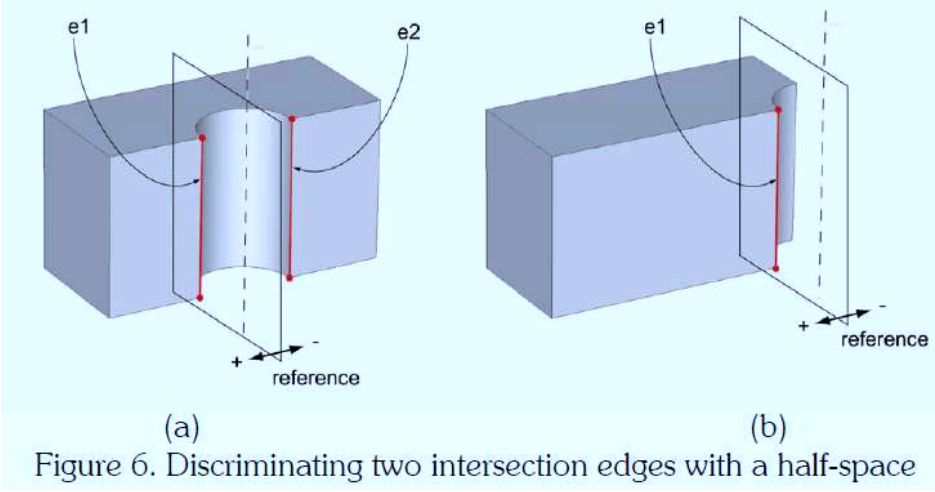


Figure 2.17 : Discrimination de 2 arêtes issues d'une intersection grâce à un demi espace, extrait de [Bid05]

Les caractéristiques sont associées à une forme paramétrique, appelée *forme canonique*, dont les frontières faces ont un nom unique et sont appelées *faces caractéristiques* (la face haute, la basse et la face formant le contour d'un cylindre par exemple). On peut également y spécifier des caractéristiques de référence, comme la direction d'un cylindre. Une classe de caractéristiques déclaratives encapsule ainsi une série de contraintes liées aux paramètres de forme et d'éléments, qui sont instanciées avec elle. On peut ensuite nommer les entités $< nomDeL'Instance > . < nomDeL'element >$ de manière unique, car la relation caractéristique - élément l'est.

— Classes de caractéristiques procédurales : contrairement aux classes de caractéristiques déclaratives, ces classes sont utilisées pour les caractéristiques associées à des arêtes, comme les chanfreins. Il propose de nommer ces arêtes : $< facesIncidentes >$ $< Discriminant >$, le discriminant incluant les faces incidentes aux extrémités.

Il qualifie cette référence aux faces faite dans la dernière classe de "plus gros challenge" de ces travaux, à cause de l'ambiguïté qui peut exister sur le nommage de celles-ci en cas d'intersection de faces non planaires.

Afin de lever cette ambiguïté, il définit des demi-espaces, de manière à ce qu'aucune paire d'arêtes ne se trouve dans le même sous espace, ou dans la même combinaison de sous-espaces. Ceux-ci sont définis par rapport à la position des faces qui s'intersectent.

Prenons comme exemple la figure 2.17. Sans demi-espace, e_1 et e_2 de la figure de

gauche et e_1 de la figure de droite s'appellent toutes les trois $< bloc.avant, rainure >$, $< bloc.haut, bloc.bas >$. Avec les demi-espaces, sur la figure de gauche, e_1 s'appelle $< bloc.avant, rainure >< bloc.haut, bloc.bas, reference.positive >$ et e_2 s'appelle $< bloc.avant, rainure >< bloc.haut, bloc.bas, reference.negative >$ et sur la figure de droite, e_1 s'appelle $< bloc.avant, rainure >< bloc.haut$, $bloc.bas, reference.positive >$.

Cette méthode à l'avantage de proposer une méthode de levée des ambiguïtés rendant unique les noms. Cependant, cette méthode fait appel à des plans, et n'est donc pas indépendante de la géométrie, ce qui peut entraîner des problèmes en cas de changements importants de celle-ci. De plus, ce plan de référence peut être difficile à définir pour les formes complexes. Enfin, cette solution ne peut marcher que pour les différentes surfaces définies dans les 3 classes, à savoir les plans, les cylindres, les sphères et les tores.

Baba Ali, 2010

Ce système de nommage, développé par Baba-Ali [BAMS09], [BA10], est basé sur les arêtes. Il choisit d'utiliser les arêtes afin d'avoir un nommage plus homogène en dimension. En effet, le fait de choisir les faces comme base du nommage, comme présenté sur les exemples précédents, vient du fait qu'elles soient de dimension $n-1$ en 3D. En effet, ces approches considèrent que les faces sont plus stables que les arêtes ou les sommets en 3 dimensions, les faces ayant forcément un ancêtre invariant, là où les arêtes et sommets peuvent résulter aussi bien de l'intersection de faces que d'arêtes. Mais ce système n'est pas homogène en dimension, les volumes étant par exemple la dimension $n-1$, et donc la plus stable en 4D. De plus, comme nous l'avons vu dans les cas précédents, le nom des arêtes repose sur ceux des faces et ceux des sommets sur ceux des arêtes. Or, le nom d'une arête prenant en compte plusieurs noms de faces, et celui des sommets plusieurs noms d'arêtes, les noms persistants des sommets peuvent rapidement reposer sur un grand nombre de faces. Ce nombre augmente encore quand la dimension du modèle augmente. Choisir l'arête comme base du nommage permet ainsi de s'appuyer sur moins d'entités invariantes que si le sommet avait été choisi, un sommet étant issu de plus d'intersections qu'une arête.

Il fait la distinction entre entités invariantes et contingentes, et nomme les arêtes invariantes :

$$[numeroEtape, nomAreteGeneratriceChemin, NomAreteGeneratriceProfil]$$

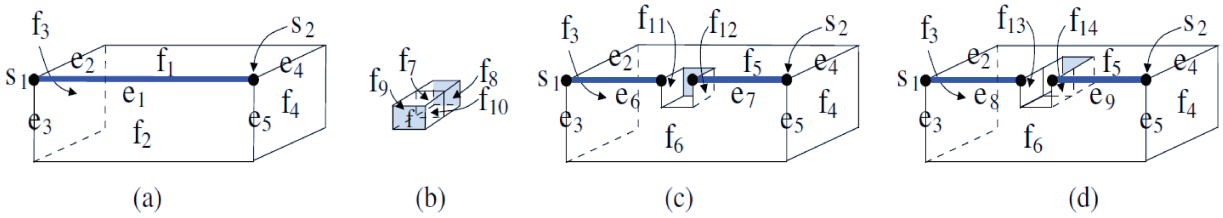


Figure 2.18 : Caractérisation d'entités contingentes chez Baba-Ali, extrait de [BA10]. a : Construction d'un premier bloc par extrusion de f_2. b : Construction d'un second bloc par extrusion de f. c : Rainure obtenue par la différence booléenne du premier et du second bloc. d : Même opération que c, mais en déplaçant le second bloc vers le fond.

et les arêtes contingentes :

$$[numeroEtape, entitesInvariantesDeDimension(n-1)Incidentes,\\ entitesInvariantesIncidentesAuPremierSommet,\\ entitesInvariantesIncidentesAuDeuxiemeSommet]$$

avec n la dimension du modèle.

Ainsi, les noms des arêtes e_1, e_6 et e_7 sur la figure 2.18 sont : $[i, \{f_1, f_2\}, \{s_1, e_2, e_3, f_3\}, \{s_2, e_4, e_5, f_4\}]$, $[i{+}2, \{f_1, f_2\}, \{s_1, e_2, e_3, f_3\}, \{f_7, coquelatérale_{rainure}, s_2\}$ et $[i{+}2, \{f_1, f_2\}, \{f_8, coquelatérale_{rainure}, s_2\}, \{s_2, e_4, e_5, f_4\}]$, i étant le numéro de l'étape qui crée la figure 2.18a.

Pour l'appariement, les arêtes invariantes sont appariées avec celles de même nom, ce qui est immédiat puisqu'elles sont invariantes. Pour les arêtes contingentes, on s'assure que les faces incidentes aient le même ancêtre invariant (l'entité invariante dont elles sont issues si elles ont été modifiées). En effet, le nom suffit à caractériser de manière unique chaque arête, mais ne suffit pas à les apparier. Si l'on observe la figure 2.18, on peut constater que e_1 peut être appariée à e_6 et e_7, qui n'ont pas le même nom. Ce lien de *recouvrement*, quand une arête initiale correspond à plusieurs arête au rejeu ou inversement, est mis en lumière par les sommets qui composent les extrémités des arêtes. Une fois déterminé quelles arêtes ont les mêmes faces incidentes, leurs sommets sont analysés (figure 2.19). Enfin, avec ces recouvrements de sommets, un double système d'équations est obtenu (figure 2.20), un pour la relation initiale $\rightarrow$ rejeu et un pour la relation rejeu $\rightarrow$ initiale. Ce système est ensuite réduit, ce qui permet d'arriver au système de la figure 2.21, une fois que les arêtes qui étaient liées dans le sens initial $\rightarrow$ rejeu (*RIR*, pour *Recouvrement Initial-Rejeu*) et

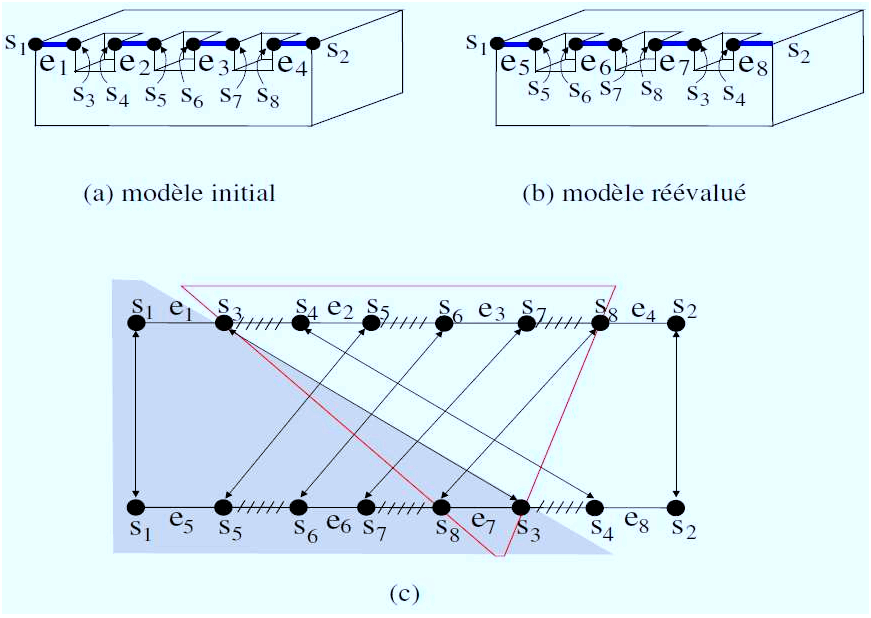


Figure 2.19 : Recouvrement de sommets, extrait de [BA10]

dans le sens rejeu → initial (*RRI*, pour *Recouvrement Rejeu-Initial*) ont été extraites.

Ainsi, sur la figure 2.20, on constate par exemple que $e_1 \rightarrow e_5, e_6, e_7$ en RIR (car ce sont les arêtes qu'on l'on peut retrouver entre les sommets formant les bords de e_1 sur 2.19), mais on n'a pas trace d'un recouvrement $e_7 \rightarrow e_1$ ou $e_6 \rightarrow e_1$ dans RRI. Seul $e_1 \leftrightarrow e_5$ est donc conservé dans 2.21.

$$RIR : \begin{cases} e_1 \longrightarrow e_5, e_6, e_7 \\ e_3 \longrightarrow e_6 \\ e_4 \longrightarrow e_7, e_8 \end{cases}$$

$$RRI : \begin{cases} e_5 \longrightarrow e_1, e_2 \\ e_6 \longrightarrow e_3 \\ e_8 \longrightarrow e_2, e_3, e_4 \end{cases}$$

Figure 2.20 : Système d'équation de résolution du recouvrement, extrait de [BA10]

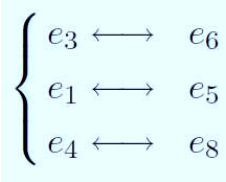


Figure 2.21 : Appariement final des arêtes dans l'exemple de Baba Ali, extrait de [BA10]

En utilisant cette méthode, des agrégats (des ensembles d'entités) peuvent ainsi être nommés et appariés, *via* les arêtes qui les composent.

Baba Ali présente une méthode de nommage générale, les arêtes en étant la base quelle que soit la dimension du modèle. Sa méthode est également presque homogène, les entités de dimension supérieures à 1 étant nommées sur le même modèle. Cependant, la manière dont est réalisé le nommage des sommets n'est pas précisée. De plus, comme il le signale dans les perspectives de sa thèse, et comme c'est le cas pour les autres travaux étudiés, le problème de l'édition d'une spécification paramétrique n'est pas abordé.

Autre approche : mise en correspondance des noms

Certaines méthodes [MH09], [CMHK12] proposent d'ajouter une couche de *mise en correspondance de noms*, afin de permettre l'échange entre systèmes de CAO distincts de tout le processus de construction (i.e. l'ensemble de la spécification paramétrique) et non uniquement le résultat de cette modélisation (i.e. le modèle géométrique final). Cette couche de mise en correspondance permet de traduire les noms dans la mesure où leur fonctionnement et les mécanismes de nommage utilisés sont la plupart du temps bien différents d'un système à l'autre. Il est à noter qu'en raison du secret industriel, il peut être assez complexe d'obtenir les informations nécessaires pour la mise en correspondance, car on ne connaît pas toujours la manière dont est réalisé le nommage en interne. Même si nos travaux ne s'intéressent pas directement à ce problème, ils sont voisins. C'est pourquoi nous l'aborderons rapidement.

Un système permettant de transférer une géométrie finie ou certaines données d'un modeleur à un autre existe déjà, *STEP* (pour *Standard for the Exchange of Product Model Data* [Owe97]).

Parmi les méthodes proposées, on peut citer les méthodes de Mun et Han [MH09] et de Cheon et al. [CMHK12], qui proposent de réaliser un langage neutre basé sur le système

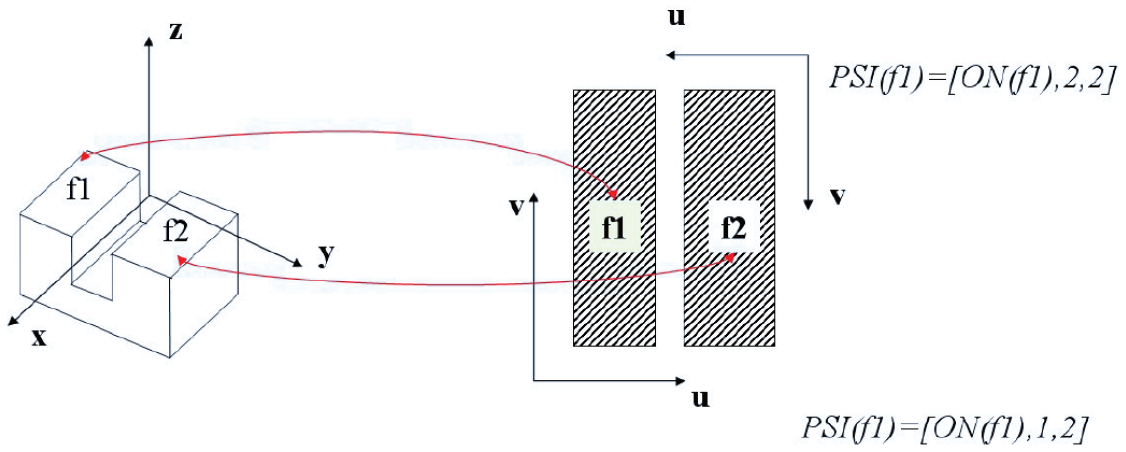


Figure 2.22 : Problème de la mise en correspondance de noms, extrait de [MH09]. À gauche, représentation de l'objet, à droite, représentation paramétrique

OSI, pour *Object Space Information*, permettant de récupérer les coordonnées des éléments dans un système de coordonnées unifié, afin de faciliter la mise en correspondance des noms. En effet, l'utilisation du système *PSI*, pour *Parametric Space Information* (proposée par Wu, et utilisé dans sa méthode et les méthodes qui en dérivent) fournit des informations dépendant des paramètres, ce qui ne permet pas la communication entre les modeleurs. En effet, si on prend l'exemple d'une face issue d'une extrusion, on peut envisager deux manière de la nommer : [Face de départ - Chemin - Face d'arrivée], ou [Face d'arrivée - Chemin - Face de départ], avec deux modeleurs différents. La figure 2.22 montre un autre exemple de ce problème, avec une même face, f_1, nommée différemment en PSI en fonction de deux modeleurs différents, qui la nomment $[ON(f_1), 1, 2]$ pour le premier et $[ON(f_1), 2, 2]$ pour le second en fonction de l'orientation de l'espace paramétrique.

On peut également citer la méthode de Tessier et Wang [TW13], qui mettent au point une méthode ontologique de mise en correspondance. On a alors :

— La mise en correspondance statique, qui parcourt les deux modèles, analyse leurs différences, et stocke ensuite des paires associées. Cependant, il nécessite la bibliothèque du modeleur pour être utilisé.

— La mise en correspondance dynamique, avec lequel, quand un élément est mis à jour, le système le compare aux éléments déjà existants dans l'autre modèle pour l'associer à l'un de ces derniers. Cependant, il nécessite de refaire le calcul à chaque mise à

jour d'une entité.

— La mise en correspondance ontologique, qui combine les forces des deux précédentes, afin de créer un système qu'il est possible de mettre à jour tout en stockant les données. Cependant, la manière d'extraire les données et de les stocker peut encore être optimisée [TW13].

Comme évoqué plus haut, la mise en correspondance de noms dans le cadre de l'échange entre modèles de CAO distincts est un problème à part entière dont la littérature fait état dans plusieurs articles. Il se trouve toutefois hors du cadre de notre travail dans la mesure où nous étudions le problème d'appariement au sein d'un unique système basé sur l'utilisation de règles de réécritures de graphes et dans le cas de l'édition d'une spécification paramétrique.

2.5 Règles de Réécriture

Les langages de *règles de réécriture*, ou *règles de transformation*, sont un des moyens d'implémenter les opérations constituant un modèle. Une règle représente la transformation d'un objet A en objet B, et s'écrit généralement $A \rightarrow B$, la partie gauche A représentant l'objet avant transformation, et la partie droite B, l'objet après transformation. Les langages basés sur les règles font partie des approches standards de la modélisation géométrique.

Les règles les plus répandues sont les *grammaires formelles* [Cho56], [Car07], définies comme :

Definition 6 (*Grammaire formelle*)

Une grammaire G est constituée des éléments :

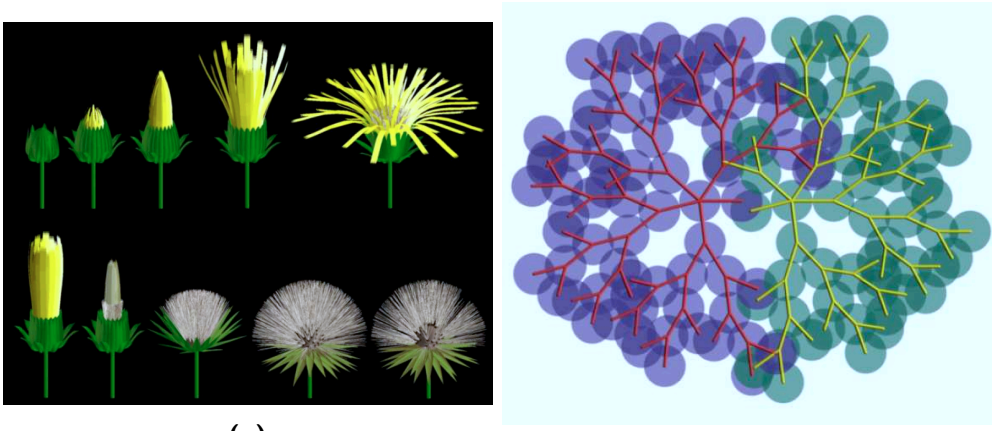
— *Un ensemble fini N de symboles non terminaux, notés conventionnellement en majuscule.*

— *Un ensemble fini T disjoint de N de symboles terminaux, notés conventionnellement en minuscule.*

— *Un symbole distinct de N, noté s, appelé axiome.*

— *Un ensemble de règles, formé d'une paire {non terminal - suite de terminaux et de non terminaux }, par exemple $\{A, ABa\}$, qui peut aussi s'écrire $A \rightarrow ABa$.*

Parmi ces grammaires formelles, on peut par exemple citer les *L-Systems* (ou L-Systèmes



(a) **(b)**

Figure 2.23 : Exemples d'utilisation des L-Systèmes. (a) Étapes du développement d'un pissenlit, [PHM93]. (b) Résultat de l'évolution de deux plantes poussant côte à côte, [MP96].

en français), un style de grammaires de génération procédurale, inventée en 1968 par le biologiste Lindenmayer [Lin68]. Ce système est basé sur le principe d'itération des applications jusqu'à satisfaire à une condition d'arrêt donnée. Il a été initialement développé pour modéliser le processus de croissance et de prolifération de plantes. Il est utilisé, par exemple, par Smith [Smi84], Rozenberg et Salomaa [RS80] ou encore Fitch et al. [FPL16]. Il est également utilisé par Prusinkiewicz et al. [PHM93] afin de visualiser le développement d'une plante ou d'une fleur à travers le temps, Mech et Prusinkiewicz [MP96], pour une étude de l'adaptation des plantes à leur environnement, ainsi que par Terraz et al. [TGMD09] afin de représenter la structure interne du bois. La figure 2.23 présente les résultats de certains de ces travaux.

On peut également utiliser des grammaires de génération procédurale en architecture pour modéliser plusieurs exemplaires d'un même bâtiment ([HMVG09], [QB15]). Dans le domaine du jeu vidéo, on les utilise également pour "meubler" plus facilement de grandes plaines, afin de faciliter l'immersion du joueur, comme par exemple dans *Arena* [Sofa] en 1994, *Oblivion* [Sofb] en 2006, ou encore *Minecraft* [Moj] en 2011. Arena et Minecraft sont deux jeux utilisant un système de génération procédurales de cartes, le premier pour pallier au problème de la place qu'aurait pris une carte de la taille de plusieurs pays sur un jeu devant tenir sur une disquette et le second, pour une expérience aléatoire du jeu.

Cependant, ce système procédural a ses limitations. Par exemple, son utilisation peut

nécessiter beaucoup d'informations pour générer ces grammaires, et chaque outil ne peut être utilisé que pour son utilisation spécifique, à cause du nombre restreint de règles de ces sytèmes [Bel12]. Ainsi, on ne peut pas utiliser l'outil de création procédurale d'un bâtiment pour créer une plante.

Nous préférons utiliser un modeleur plus extensible. Pour cela nous utiliserons un système de règles appartenant aux *règles de transformation de graphes* [EEP06]. Ce système a déjà été utilisé en combinaison avec les modèles topologiques [Bel12], [Pou09], ce qui correspond à notre cadre de travail. Il permet de définir tout type d'opérations, sans que leur utilisation ne soit dans le contexte d'un domaine d'application précis. La manière de représenter une règle devient $L \rightarrow R$, avec L le graphe filtré par la règle, et R le motif transformé par la règle.

Jerboa

Afin de réaliser notre système de nommage, nous utilisons la bibliothèque Jerboa [dP CP] [BALGB14], qui permet de produire des règles de transformation de graphe. Elle est conçue pour assister au développement de modeleurs dédiés à un certain type d'application.

Jerboa crée des règles indépendantes de l'utilisation finale, même si elles sont utilisées par un modeleur dédié à un certain type d'applications. Ainsi, une règle créée dans un modeleur dédié à l'architecture pourra tout aussi bien servir dans un modeleur dédié à la biologie. De plus, les règles n'y sont pas codées par l'utilisateur, mais dessinées à l'aide d'un éditeur de règles, même si l'utilisateur peut ensuite affiner ces règles grâce au code, en précisant par exemple des pré-conditions nécessaires à leur application. Enfin, les règles Jerboa sont conçues pour transformer les G-Cartes, et chaque règle préserve l'intégrité de la G-carte qu'elle modifie. Cela signifie que la règle ne pourra pas transformer la G-Carte en un élément qui ne respecte pas la définition d'une G-Carte. En effet, l'éditeur préviendra l'utilisateur si une règle ne permet pas d'assurer cette intégrité. Il lui indiquera également quelles erreurs corriger pour que la règle soit valable.

La partie gauche de la règle permet de filtrer le type d'orbite qui sera modifié. Ainsi, la figure 2.24 présente la règle de triangulation. La partie gauche indique que l'on filtre le type d'orbite $\langle 0, 1 \rangle$, correspondant à une face. Appliquons-la sur le brin a_0 de la figure 2.25a. Notons que le choix du brin est arbitraire, la règle s'appliquant sur une orbite, n'importe

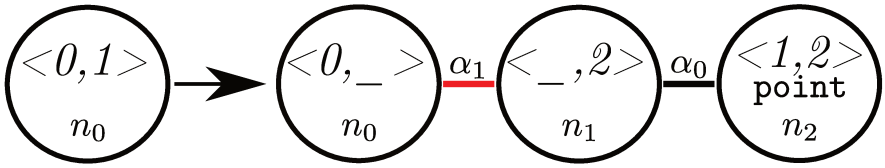


Figure 2.24 : Règle Jerboa de triangulation

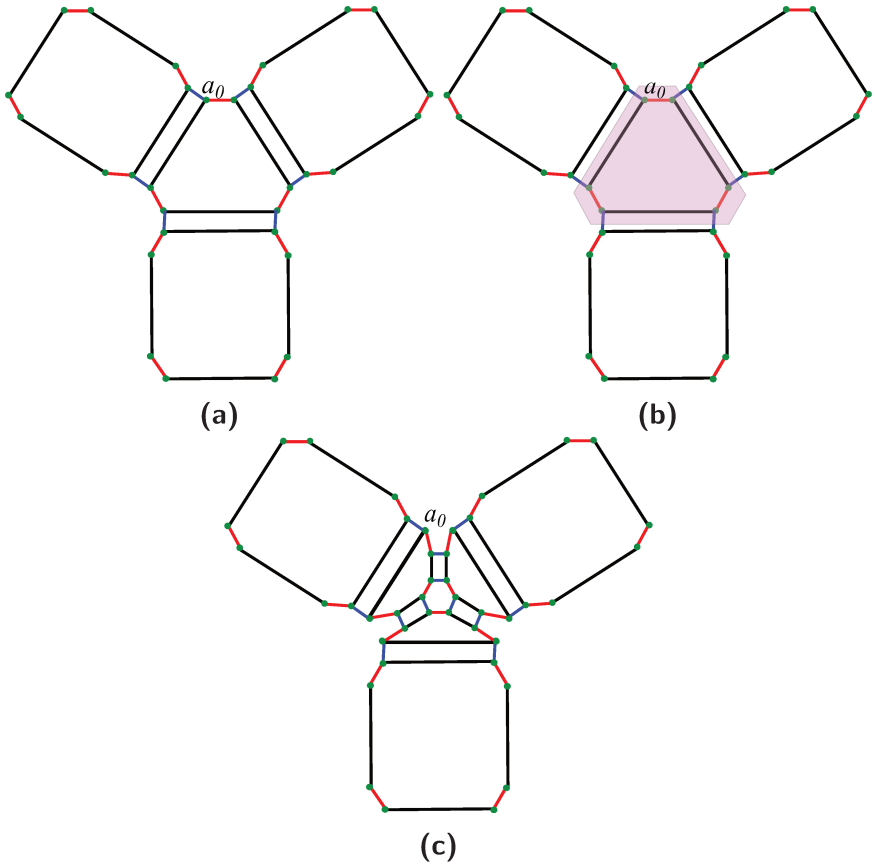


Figure 2.25 : Triangulation d'une face d'un modèle. a : modèle auquel appartient la face. b : mise en évidence de la face à trianguler. c : résultat de la triangulation.

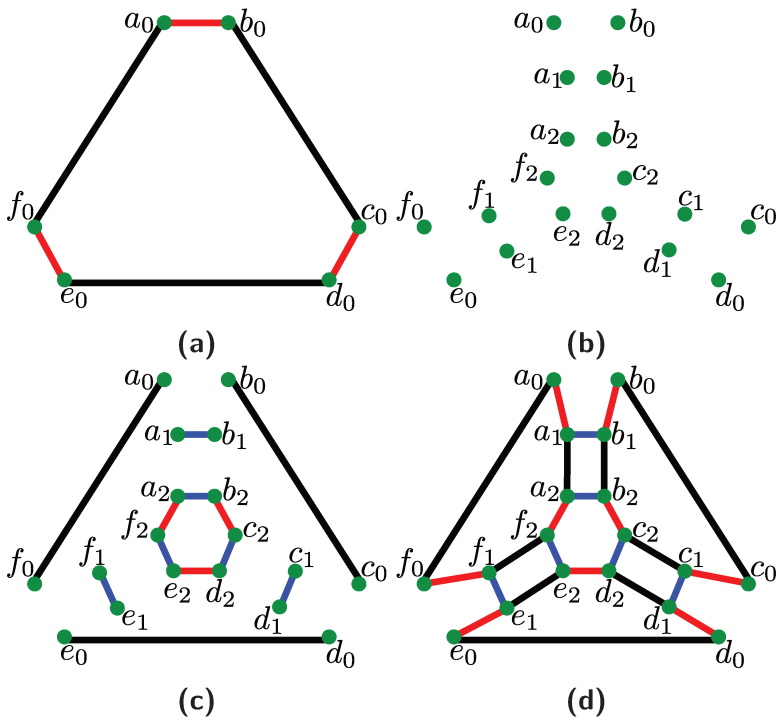


Figure 2.26 : Étapes successives de l'application de la règle de triangulation. a : motif avant filtrage. b : copies des brins existants. c : ajout des liaisons entre brins issus du même nœud. d : ajout des liaisons entre brins issus de différents nœuds.

quel brin de cette orbite permet de la désigner. La figure 2.25b montre l'orbite face $a_0.\langle 0,1\rangle$ ainsi filtrée par la règle. La figure 2.25c montre l'orbite triangulée après transformation. Les autres faces n'ont pas été affectées. Pour expliquer les étapes de l'application d'une règle, nous allons nous concentrer uniquement sur la face $a_0.\langle 0,1\rangle$ (figure 2.26).

La partie droite de la règle nous permet d'indiquer comment réécrire le motif filtré, c'est-à-dire, ici, la face triangulaire liée à a_0. Notons que lorsqu'un nœud n_0 est présent à la fois à gauche et à droite, cela indique que les brins qu'il filtre sont présents à la fois avant l'application de la règle et après. Ce nœud est alors *réécrit*.

Les brins filtrés par le nœud n_0 sont tout d'abord copiés. Ici, on a à droite n_0, n_1 et n_2. On réalise donc deux copies de ces brins, n_0 étant déjà présent dans la partie gauche de la règle. Les brins $a_0, ..., f_0$ créent ainsi les brins $a_1, ..., f_1$ et $a_2, ..., f_2$ par copie. Le résultat de cette première étape est présenté sur la figure 2.26b. Comme indiqué précédemment, les brins filtrés par n_0 (a_0 ...f_0) sont présents à la fois avant et après l'application de la

triangulation.

Le deuxième étape consiste à ajouter les liaisons entre les brins issus d'un même nœud. De nouveau, le nœud de la partie gauche nous donne des indications. En effet, les types d'orbites de la partie droite en sont des réécritures, qui décrivent comment lier les brins issus d'un même nœud. Ici, on sait que l'on y filtre le type d'orbite $\langle 0, 1\rangle$ avec n_0. Celui-ci est réécrit, dans la partie droite, en $\langle 0, \_\rangle$ pour n_0, $\langle \_, 2\rangle$ pour n_1 et $\langle 1, 2\rangle$ pour n_2. Le symbole _ indique qu'une liaison est supprimée par la réécriture. Prenons le cas des brins issus de n_0. Avant application de la règle, a_0 et b_0 étaient liés par α_1 et b_0 et c_0 par α_0. Après l'application, $\langle 0, 1\rangle$ étant réécrit en $\langle 0, \_\rangle$, a_0 et b_0 ne sont plus liés, 1 étant réécrit en _, et b_0 et c_0 sont liés par α_0, 0 étant réécrit en 0. Le résultat de cette deuxième étape est présenté sur la figure 2.26c.

Enfin, il faut lier entre eux les brins issus de nœuds différents. Pour cela, on relie deux brins issus de nœuds différents par les liaisons indiquées sur la partie droite de la règle. Ces liaisons s'appliquent entre deux brins issus d'une même copie. Ici, les brins issus de n_0 sont liés à ceux issus de n_1 par α_1, et ces derniers sont liés à ceux issus de n_2 par α_0. Ainsi, a_0 est lié à a_1 par α_1 et a_1 est lié à a_2 par α_0, b_0 est lié à b_1 par α_1 et b_1 est lié à b_2 par α_0... Le résultat de cette deuxième étape est présenté sur la figure 2.26d.

Il existe déjà de nombreuses utilisations de Jerboa, dans des domaines variés comme l'architecture [HMGB09], la géologie [GAB+16] ou la modélisation d'éléments physiques [BSBAM17]. Cependant, Jerboa ne permet pas encore la réévaluation rapide de modèles grâce au changement de paramètres et à l'édition de spécification permis par la modélisation paramétrique.

Notre but ici est double :

— d'une part, intégrer les spécifications paramétriques dans Jerboa, afin d'étendre son domaine d'application,

— d'autre part, exploiter les avantages des règles de transformation de graphe pour créer un nouveau système de nommage, et ainsi étendre le champ d'étude du nommage persistant aux modélisations basées sur ces règles de transformation de graphe.

Let's get back to business

Mulan - Disney

3

Outils permettant de rejouer une spécification

Sommaire

L'objectif de notre travail est de réaliser un nommage persistant robuste et un système de rejeu permettant d'ajouter, supprimer ou déplacer des opérations dans une spécification paramétrique. L'utilisation des règles de transformation de graphe Jerboa a un but double :

— permettre l'utilisation de Jerboa dans le cadre de la modélisation paramétrique,

— utiliser les mécanismes de Jerboa pour étendre le nommage.

Afin de présenter notre système de nommage et de rejeu, nous allons nous appuyer sur

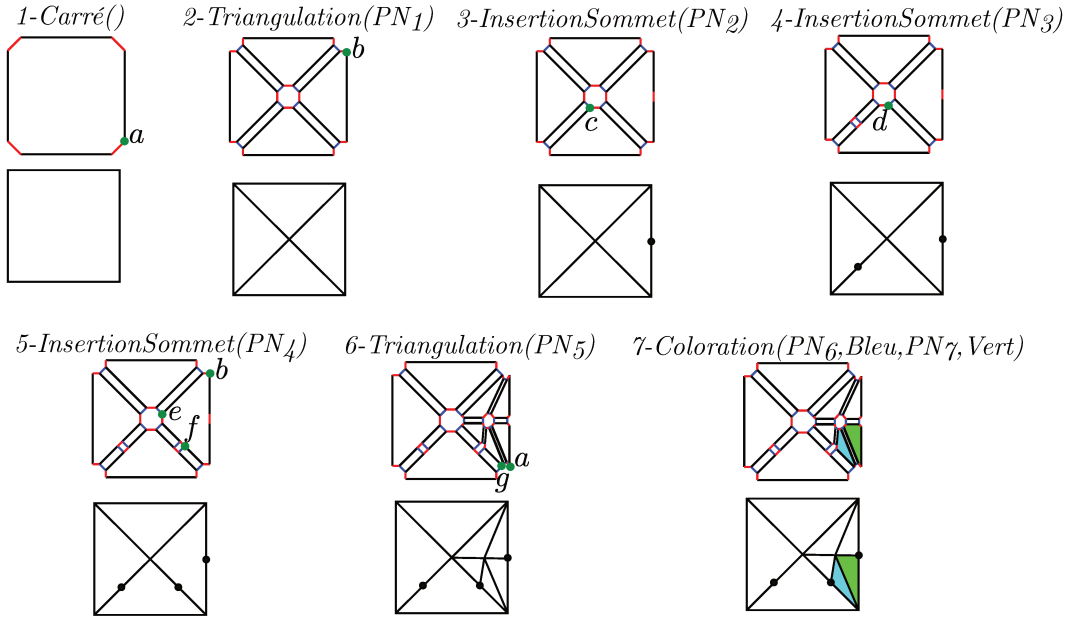


Figure 3.1 : Exemple "Fil Rouge" permettant de dérouler notre mécanisme de rejeu

l'exemple "fil rouge" présenté sur la figure 3.1. Pour plus de facilité de lecture, chaque modèle est présenté selon 2 représentations, la représentation G-Carte, et en dessous la représentation géométrique correspondante. Au-dessus de cet ensemble se trouve l'opération qui a créé cette étape. Par exemple, on trouve $1 - Carre()$, l'opération, puis en-dessous, la représentation G-Carte du carré, et enfin, encore en dessous, sa représentation géométrique.

Une opération fait en réalité appel à une règle Jerboa, c'est pourquoi, dans la suite du mémoire, il sera possible d'utiliser le terme "opération" pour "règle Jerboa". Les différents brins indiqués sont ceux sur lesquels reposent la prochaine opération. Par exemple, le paramètre PN_1 de $2 - Triangulation$ permet de désigner la face associée au brin a. Il s'agit en fait du nom persistant de la face qui doit être triangulée. Ce nommage persistant est détaillé dans la section 3.1.2.

La figure présente une suite de 7 opérations. On crée tout d'abord un carré, puis on le triangule. On réalise ensuite trois insertions de sommet successives sur les arêtes du modèle, avant de trianguler sa face droite. Enfin, on colore deux des faces issues de cette triangulation, une en bleu et une en vert.

Dans ce troisième chapitre, nous mettons en place les différents outils permettant de réaliser le rejeu. Pour cela, nous présentons tout d'abord notre système de nommage per-

sistant, puis notre système de *journaux de bord*, permettant de prévoir les actions d'une règle, et notre système de *journaux d'historique*, qui utilisent les deux éléments précédents afin de permettre de suivre les différentes modifications qui affectent les entités du modèle. La manière de les utiliser est ensuite détaillée dans le chapitre 4.

3.1 Nommage persistant

Comme nous l'avons évoqué dans la section 2.4, le nommage persistant est essentiel à la réalisation du rejeu.

Notre système de nommage se base sur les brins des G-Cartes et sur les règles Jerboa pour permettre un nommage non ambigu. Nous allons maintenant détailler la manière dont nous le mettons en place, tout d'abord en expliquant comment identifier les brins, puis comment se servir de ce premier identifiant pour nommer les entités.

3.1.1 Nommage des brins : l'identifiant persistant

Les brins sont l'entité de base d'une G-Carte, la seule qui ne puisse pas être subdivisée en d'autres entités. Nous décidons donc de nous en servir comme point de départ de notre nommage. Ce nom s'appuie également sur les règles Jerboa. En effet, chaque brin résultant de l'application d'une règle ne peut être que :

— crée par réécriture ou par copie d'un autre brin comme dans le cas de la triangulation (figure 3.2a) ;
— ou bien créé *ex-nihilo* par une règle de création comme par exemple, la règle de création d'un carré (figure 3.2b), où il n'y a pas de partie gauche.

Un brin ne peut ainsi être caractérisé que par une seule suite de règles et de nœuds, et ce de manière unique. Les brins sont nommés grâce à un *identifiant persistant* noté PI.

Le système d'identification persistante mis en place consiste en une suite d'associations $\{numeroEtape - noeud\}$, permettant de garder trace de chaque opération ayant affecté le brin. L'identifiant est ensuite mis à jour à chaque nouvelle opération appliquée sur une orbite à laquelle le brin appartient. En notant i le numéro de l'étape de l'opération, la mise à jour de l'identifiant se fait de la manière suivante :

— Brin créé *ex-nihilo* : le brin prend automatiquement comme identifiant $\{i - noeud\}$.
— Brin réécrit par une règle : l'identifiant est mis jour en y ajoutant $\{i - noeud\}$.

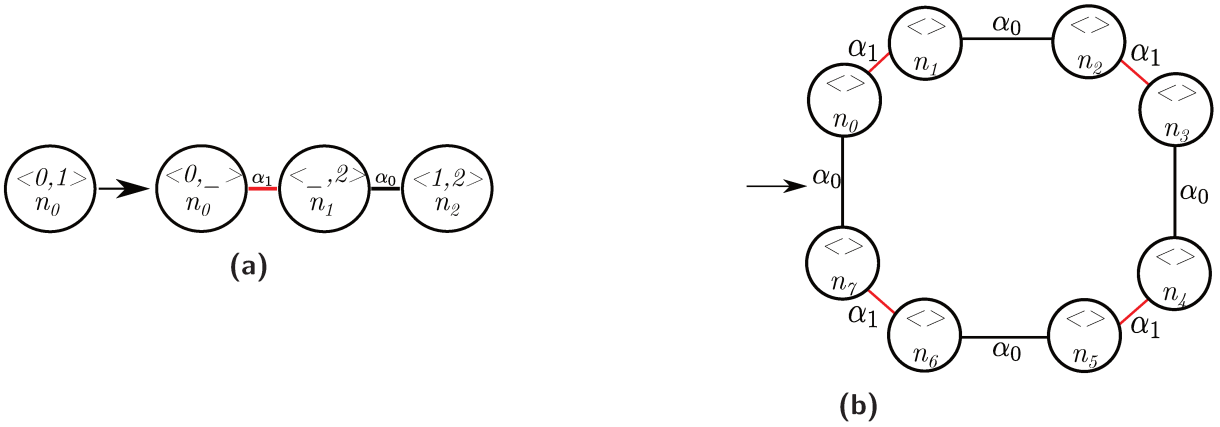


Figure 3.2 : a : Règle de triangulation, transformant une face. b : Règle de création de carré.

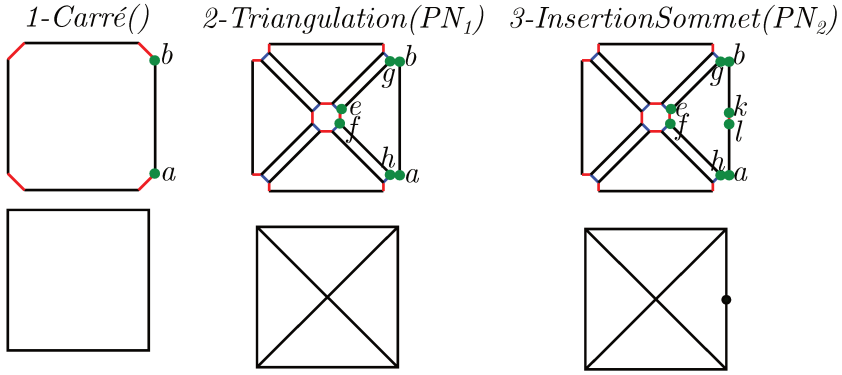


Figure 3.3 : Début du modèle paramétrique fil rouge pour l'application de l'identification et du nommage persistants

— Brin créé par copie : l'identifiant est créé à partir de l'identifiant du brin copié à une étape $j < i$, auquel on ajoute $\{i - noeud\}$.

On a donc la définition suivante pour nos identifiants persistants :

Definition 7 (*Identifiant Persistant*)

Soit N le nombre d'opérations de la spécification paramétrique.
L'identifiant persistant PI_b du brin b est défini comme la liste des $1 \leq m \leq N$ éléments $\{e_1, ..., e_m\}$ caractérisant l'ensemble des opérations et des nœuds ayant réécrit b, où chaque élément $e_i = \{x - n_y\}$ définit le nœud n_y de la règle associée à l'opération numérotée x ayant filtré le brin b, avec $1 \leq x \leq N$ et $1 \leq i \leq m$.

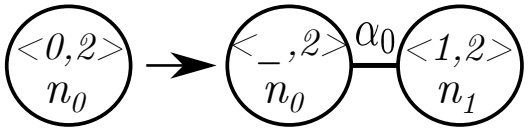


Figure 3.4 : Opération d'insertion de sommet dans une arête

Le figure 3.3 nous permet d'appliquer ce système d'identification persistante des brins sur un exemple.

À l'étape 1, les brins a et b sont créés. L'opération 1-Carré crée ces brins *ex-nihilo*. L'opération porte le numéro 1, et a est créé par n_4 et b par n_3. On aura donc $PI_a = \{1 - n_4\}$ et $PI_b = \{1 - n_3\}$, PI désignant l'identifiant persistant.

À l'étape 2, tous les brins de la face carrée sont affectés par la triangulation (figure 3.2a). Notamment, a et b étant filtrés par n_0, leurs identifiants sont mis à jour et deviennent : $PI_a = \{1 - n_4; 2 - n_0\}$ et $PI_b = \{1 - n_3; 2 - n_0\}$. D'autres brins, comme e, sont créés par copie ; e est créé *via* le nœud n_2 par copie de b. On récupère donc l'identifiant de b à l'étape précédente, $\{1 - n_3\}$, que l'on complète avec les informations issues de l'étape 2, à savoir le fait que e est issu d'une copie par n_2, pour obtenir : $PI_e = \{1 - n_3; 2 - n_2\}$.

Enfin, à l'étape 3, on insère un sommet sur l'arête à laquelle appartiennent a et b ; e n'est pas affecté, son identifiant ne change donc pas. Les identifiants de a et b sont eux mis à jour, ainsi que les identifiants des 2 brins k et l créés par l'opération. La figure 3.4 présente la règle d'insertion de sommet, et celle-ci permet d'identifier les deux brins : $PI_a = \{1 - n_4; 2 - n_0; 3 - n_0\}$ et $PI_b = \{1 - n_3; 2 - n_0; 3 - n_0\}$. Les deux brins créés sont identifiés $PI_l = \{1 - n_4; 2 - n_0; 3 - n_1\}$ pour la copie de a et $PI_k = \{1 - n_3; 2 - n_0; 3 - n_1\}$ pour la copie de b.

Le fonctionnement des règles Jerboa garantit l'unicité et la non-ambiguïté des identifiants persistants. En effet, chaque brin ne peut être la réécriture ou la copie que d'un seul brin et leurs identifiants diffèrent immédiatement, car ils sont issus de nœuds différents. Cependant, les opérations étant généralement appliquées sur des entités topologiques précises (i.e des orbites précises) et non sur des brins, il faut également pouvoir identifier ces entités (section 2.3) afin de les retrouver lors du rejeu de la spécification. L'identification des brins est un premier pas vers le nommage des entités. En effet, un brin ne pouvant pas appartenir à deux orbites de même type, les noms persistants d'entités sont *a priori* bien uniques et donc non ambigus s'ils se basent sur les identifiants persistants des brins.

3.1.2 Nommage des entités : le nom persistant

La seconde étape pour la réalisation de notre nommage est de donner un nom aux entités. Celui-ci est appelé *nom persistant*.

Il est créé quand une opération utilise l'entité dans la spécification paramétrique, et repose sur les identifiants persistants des brins composant l'entité. Ce nom est créé de la manière suivante :

— Dans un premier temps, on parcourt tous les brins de l'entité pour récupérer leur identifiant persistant.

— Dans un deuxième temps, on optimise cette liste, afin de la réduire. Pour cela, on sélectionne un premier identifiant, et on élimine tous les identifiants identiques aux nœuds près (c'est-à-dire, les identifiants des brins "affectés" créés *ex nihilo*, réécrits ou créés par copie par exactement les mêmes opérations). On sauvegarde ce premier identifiant, puis, s'il reste des identifiants dans la liste, on sélectionne à nouveau le premier, et on répète ce processus jusqu'à ce que la liste soit vide. Ainsi, la liste d'identifiants sélectionnés rend compte de toutes les opérations ayant "affecté" les brins de l'entité. Un exemple est présenté plus bas dans cette section.

  Par convention, on commencera toujours cette sélection en prenant comme premier identifiant de comparaison celui du brin sélectionné par l'utilisateur.

— Enfin, dans un troisième temps, on ajoute à cette liste d'identifiants sélectionnés le type d'orbite, correspondant à l'entité à nommer. Rappelons qu'un brin ne pouvant appartenir qu'à une seule orbite d'un type donné, ajouter cette information permet de rendre le nom non ambigu.

On a donc la définition suivante pour nos noms persistants :

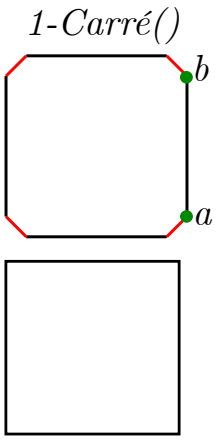


Figure 3.5 : Application du nommage persistant : zoom sur l'étape 1

Definition 8 (*Nom Persistant*)

Soit N le nombre d'opérations de la spécification paramétrique.
Le nom persistant PN_E de l'entité E est défini comme une paire composée :

- *d'une liste de $1 \leq q \leq N$ identifiants persistants $\{PI_1, ..., PI_q\}$ telle que :*
  - *$\forall PI_j$ avec $1 \leq j \leq q$, PI_j est l'identifiant d'un brin de E*
  - *$\forall PI_j$ avec $1 \leq j \leq q$, $\exists e_i = \{x - n_y\} \in PI_j / \nexists e_k = \{x - n_z\} \in \{PI_1, ..., PI_q\} \setminus PI_j$, i.e. chaque PI_j référence au moins une opération qui n'est référencée par aucun autre PI*
  - *pour toute règle associée à l'opération x ayant réécrit un brin de E, $\exists e_i = \{x - n_y\} / e_i \in PI_j$ et $PI_j \in \{PI_1, ..., PI_q\}$, i.e. l'ensemble de ces PI référence toutes les opérations ayant affecté un brin de E.*
- *d'un type d'orbite $\langle o \rangle$, définissant la dimension topologique de l'entité E.*

PN_E est noté $\{PI_1 ... PI_q\}.\langle o \rangle$.

Reprenons l'exemple de la figure 3.3. La première opération, 1-Carré, ne s'applique sur aucune entité, il n'y a donc pas de nom persistant pour cette étape.

La deuxième opération, $2-Triangulation$, s'applique sur la face carrée (figure 3.5). Celle ci est composée de 8 brins, identifiés : $\{1 - n_0\}$, $\{1 - n_1\}$, $\{1 - n_2\}$, $\{1 - n_3\}$, $\{1 - n_4\}$, $\{1 - n_5\}$, $\{1 - n_6\}$ et $\{1 - n_7\}$. Le brin sélectionné par l'utilisateur, a (voir figure 3.1), porte l'identifiant $\{1 - n_4\}$. Les autres brins présents dans la liste ne comportent tous qu'une référence à l'étape 1. On les élimine donc, pour ne garder que PI_{a1}. L'entité étant

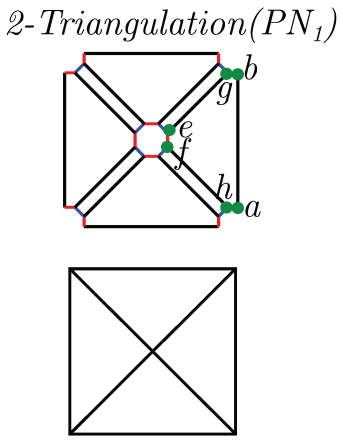


Figure 3.6 : Application du nommage persistant : zoom sur l'étape 2

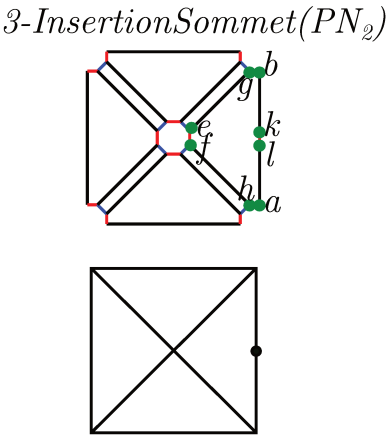


Figure 3.7 : Application du nommage persistant : zoom sur l'étape 3

une face (soit l'orbite $\langle 0, 1\rangle$ en 2D), son nom persistant est donc finalement : $PN_1 = \{\{1 - n_4\}\}.\langle 0, 1\rangle$.

La troisième opération est une insertion de sommet sur l'arête composée de a et b (figure 3.6) dont les identifiants sont respectivement $PI_{a2} = \{1 - n_4; 2 - n_0\}$ et $PI_{b2} = \{1 - n_3; 2 - n_0\}$. L'utilisateur a sélectionné b pour désigner la face. À nouveau, les deux brins ont été affectés par les mêmes opérations, on garde donc uniquement PI_{b2} pour créer PN_2, ce qui donne $PN_2 = \{\{1 - n_3; 2 - n_0\}\}.\langle 0, 2\rangle$. Le résultat apparaît sur la figure 3.7.

Enfin, imaginons qu'à une quatrième étape nous voulions appliquer une opération sur la face composée de a, b, e, f, g, h, k et l.

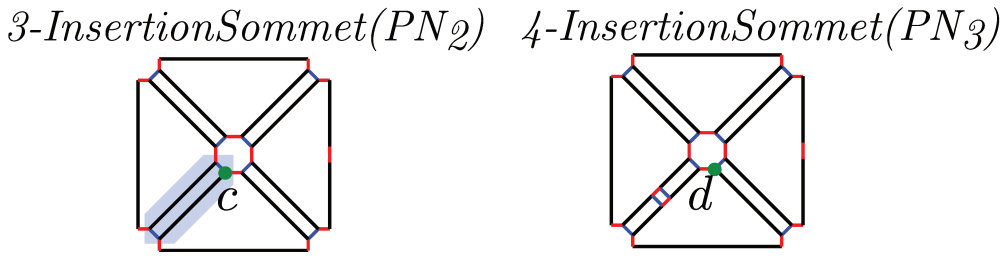


Figure 3.8 : Nommage persistant du fil rouge : PN_3

Leurs identifiants sont à la fin de l'étape 3 : $PI_{a3} = \{1 - n_4; 2 - n_0; 3 - n_0\}$, $PI_{b3} = \{1 - n_3; 2 - n_0; 3 - n_0\}$, $PI_{l3} = \{1 - n_4; 2 - n_0; 3 - n_1\}$, $PI_{k3} = \{1 - n_3; 2 - n_0; 3 - n_1\}$, affectés par l'opération 3 d'insertion d'arête, et $PI_{e2} = \{1 - n_3; 2 - n_2\}$, $PI_{g2} = \{1 - n_3; 2 - n_1\}$, $PI_{f2} = \{1 - n_4; 2 - n_2\}$ et $PI_{h2} = \{1 - n_4; 2 - n_1\}$ (voir section 3.1.1 pour le détail de la construction de ces identifiants). Sélectionnons par exemple le brin e. Son identifiant étant $\{1 - n_3; 2 - n_2\}$, les identifiants PI_{f2}, PI_{g2} et PI_{h2} sont éliminés, car ils tiennent également compte des opérations 1 et 2 uniquement.

Il reste donc les identifiants PI_{a3}, PI_{b3}, PI_{k3} et PI_{l3}. On sélectionne le premier, $PI_{a3} = \{1 - n_4; 2 - n_0; 3 - n_0\}$, ce qui élimine les autres qui comprennent également uniquement des références aux opérations 1, 2 et 3. Le nom de cette face hypothétiquement désignée serait donc :
$PN_{exemple} = \{\{1 - n_3; 2 - n_2\}; \{1 - n_4; 2 - n_0; 3 - n_0\}\}.\langle 0, 1\rangle$.

Reprenons l'exemple fil rouge en entier. Comme nous venons de le voir, PN_1 est formé grâce à PI_{a1} et PN_2 grâce à PI_{b2}. Sur les figures suivantes, permettant de comprendre comment sont formés les 5 autres noms persistants, on utilise un système de surlignage avec code couleur afin de mettre en avant les opérations ayant affecté les brins. Deux brins surlignés avec la même couleur ont été affectés par les mêmes opérations.

La partie gauche de la figure 3.8 présente le modèle avant la deuxième insertion de sommet sur une arête, ici la diagonale en bas à gauche. On peut voir que les 4 brins qui composent l'arête ont été affectés par la même suite d'opérations. Celles-ci sont la création du carré, opération 1, et la triangulation, opération 2. L'utilisateur ayant sélectionné le brin c, c'est son identifiant persistant que nous utiliserons pour créer PN_3, à savoir $PI_{c2} = \{1 - n_6; 2 - n_2\}$.

La partie gauche de la figure 3.9 présente le modèle avant la troisième insertion de

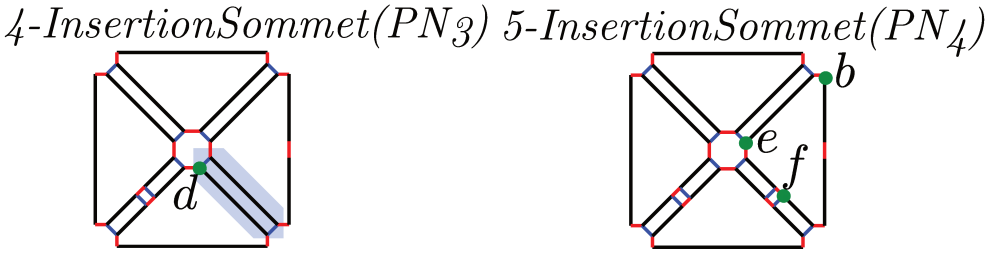


Figure 3.9 : Nommage persistant du fil rouge : PN_4

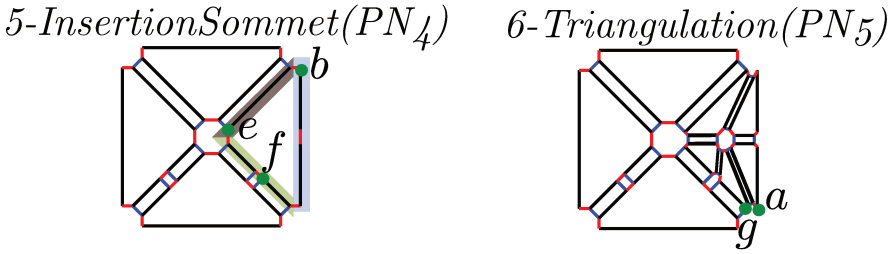


Figure 3.10 : Nommage persistant du fil rouge : PN_5

sommet sur une arête, ici la diagonale en bas à droite. On constate que les 4 brins qui composent l'arête sont affectés par la même suite d'opérations. Celles-ci sont, à nouveau, la création du carré, opération 1, et la triangulation, opération 2. L'utilisateur ayant sélectionné le brin d, c'est son identifiant persistant que nous utiliserons pour créer PN_4, à savoir $PI_{d2} = \{1 - n_5; 2 - n_2\}$.

La partie gauche de la figure 3.10 présente la modèle avant la deuxième triangulation, appliquée à la face droite du modèle. On voit que cette fois-ci, il y a 3 codes couleur pour les brins. Le premier groupe, en bleu, est affecté par 1-Carré, 2-Triangulation et 3-InsertionDroite. Le deuxième, en mauve, est affecté par 1-Carré et 2-Triangulation. Enfin, le troisième, en vert, est affecté par 1-Carré, 2-Triangulation et 5-InsertionSommet. L'utilisateur ayant sélectionné le brin e, c'est son identifiant persistant qui servira à tenir compte des opérations 1 et 2, $PI_{e2} = \{1 - n_3; 2 - n_2\}$. Il faut également un brin venant de chacun des deux autres groupes, afin de tenir compte de toutes les opérations ayant affecté les brins de la face. Les brins étant interchangeables au sein d'un même groupe, comme nous l'avons vu précédemment (on prend systématiquement le premier de la liste), on sélectionne f pour le groupe vert et b pour le groupe bleu. PN_5 est donc composé de $PI_{e2} = \{1 - n_3; 2 - n_2\}$, $PI_{f5} = \{1 - n_4; 2 - n_1; 5 - n_1\}$ et $PI_{b3} = \{1 - n_3; 2 - n_0; 3 - n_0\}$. Notons que PI_b est uti-

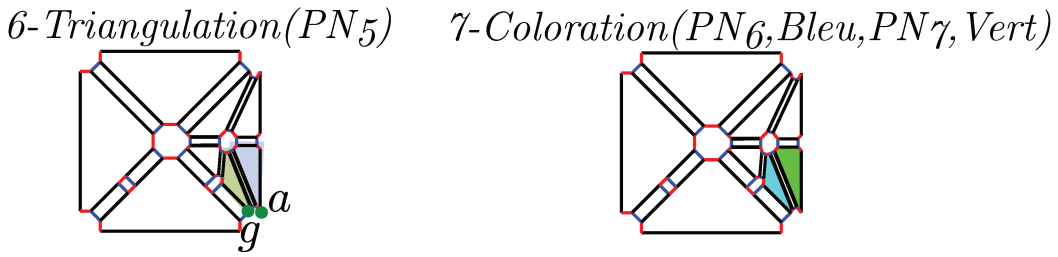


Figure 3.11 : Nommage persistant du fil rouge : PN_6 et PN_7

lisé dans deux noms persistants, PN_5 et PN_2, mais à des étapes différentes. Un seul brin suffit à désigner une face (ou n'importe quel autre type d'orbite). Cependant, en sélectionner plusieurs permet de rendre compte de toutes les opérations l'ayant affectée au cours de sa construction et de son évolution.

Enfin, la partie gauche de la figure 3.11 présente le modèle avant les deux colorations finales, appliquées à deux des faces issues de la triangulation précédente. On observe que cette fois-ci, il y a 2 codes couleur pour les brins. Chacun de ces codes couleur correspond à une face. Les noms persistants seront donc composés d'un seul brin. L'utilisateur ayant choisi g pour la coloration bleue et a pour la coloration verte, on utilise leur identifiant pour créer PN_6 et PN_7, respectivement. PN_6 sera donc composé de $PI_{g6} = \{1 - n_4; 2 - n_1; 5 - n_0; 6 - n_0\}$, et PN_7 de $PI_{a6} = \{1 - n_4; 2 - n_0; 3 - n_0; 6 - n_0\}$.

Au final, les différents noms de notre exemple fil rouge sont présentés sur le tableau 3.1.

PN	Brin(s)	Type d'orbite	Nom
PN_1	a	$\langle 0,1\rangle$	$\{\{1 - n_4\}\}.\langle 0,1\rangle$
PN_2	b	$\langle 0,2\rangle$	$\{\{1 - n_3; 2 - n_0\}\}.\langle 0,2\rangle$
PN_3	c	$\langle 0,2\rangle$	$\{\{1 - n_6; 2 - n_2\}\}.\langle 0,2\rangle$
PN_4	d	$\langle 0,2\rangle$	$\{\{1 - n_5; 2 - n_2\}\}.\langle 0,2\rangle$
PN_5	b, e, f	$\langle 0,1\rangle$	$\{\{1 - n_3; 2 - n_0; 3 - n_0\}; \{1 - n_3; 2 - n_2\}; \{1 - n_4; 2 - n_1; 5 - n_1\}\}.\langle 0,1\rangle$
PN_6	g	$\langle 0,1\rangle$	$\{\{1 - n_4; 2 - n_1; 5 - n_0; 6 - n_0\}\}.\langle 0,1\rangle$
PN_7	a	$\langle 0,1\rangle$	$\{\{1 - n_4; 2 - n_0; 3 - n_0; 6 - n_0\}\}.\langle 0,1\rangle$

Table 3.1 : Noms persistants correspondant à l'exemple fil rouge

Ces noms persistants d'entités sont donc bien uniques et non ambigus. Il est à noter

que contrairement aux identifiants persistants, mis à jour à chaque fois qu'une opération affecte le brin, le nom persistant n'évolue pas, et est fixé au moment où l'entité est utilisée.

3.1.3 Résumé

En résumé, notre système de nommage est composé de deux éléments, l'identifiant persistant de brins et le nom persistant d'entités, ce dernier se construisant à partir du premier. Le tableau 3.2 rappelle les principales caractéristiques de ce système.

	Identifiant persistant de brins	Nom persistant d'entités
Caractérise :	Les brins	Les entités
Basé sur :	Les numéros des opérations et des nœuds	Les identifiants persistants et les types d'orbites
Évolution :	Mis jour à chaque opération affectant le brin	N'évolue pas, créé au moment de l'utilisation de l'entité par une opération
Utilisé pour :	Créer les noms persistants	Identifier les entités pour les apparier

Table 3.2 : Résumé du système de nommage persistant

Ce système permet un nommage homogène, toutes les entités étant nommées de la même manière, et général, les noms étant basés sur les brins, qui sont l'entité de base d'une G-Carte, peu importe sa dimension.

Concernant la position de ce système par rapport aux concepts communs, l'utilisation de règles Jerboa élémentaires permet de ne pas faire la distinction entre entités invariantes et contingentes, toutes les entités étant modifiées et nommées de la même manière. L'architecture Modèle-Nom-Spécification est bien respectée, les opérations de la spécification faisant en fait appel à des règles Jerboa. Enfin, un système d'historique est intégré, porté par les identifiants persistants, et, grâce à eux, par les noms persistants. Il est complété par un système de journal d'historique, que nous développerons dans la section 3.3.

3.2 Journaux de bord

Les noms persistants nous permettent d'identifier les entités. Nous allons maintenant voir comment les utiliser pour réaliser l'appariement de ces mêmes entités.

Dans un premier temps, il nous faut connaître quelles modifications affectent les entités, information non-contenue dans le nom persistant. En effet, il s'agit d'un critère important devant être pris en compte dans le calcul d'appariement et permettant de savoir si l'on cherche zéro, un ou plusieurs appariements pour une même entité. Imaginons par exemple qu'une face soit colorée au jeu initial et qu'au rejeu cette même face se retrouve scindée en deux. Détecter la modification de la face intiale (sa scission) permettra de l'apparier avec les deux faces issues de la scission et donc d'appliquer correctement la coloration sur ces dernières. Pour cela, on crée un journal de bord de règle, qui permettra d'indiquer les modifications subies par les entités qu'elle affecte. Celui-ci est créé en même temps que la règle à laquelle il est associé.

Nous verrons au chapitre 4 qu'il est possible de paramétrer l'appariement en fonction des types de modifications et des opérations. On peut, par exemple, choisir de ne pas réaliser un appariement quand une entité n'est pas créée au rejeu (c'est-à-dire, si par exemple l'opération qui la créait a été supprimée).

Mais voyons dans un premier temps les différents types de modification. On en distingue six :

— Création : une entité est créée. Cela signifie que tous les brins qui la composent sont créés par l'opération. C'est le cas de toutes les orbites de la figure 3.12a. Par exemple, si une entité est créée au rejeu mais ne l'était pas au jeu initial, on sait qu'elle n'aura pas d'appariement.

— Suppression : une entité est supprimée. Tous les brins qui la composent sont supprimés (par exemple le sommet vert sur la figure 3.12a, supprimé sur la figure 3.12b). Par exemple, si une entité est supprimée au rejeu mais ne l'était pas au jeu initial, on sait qu'elle n'aura plus d'appariement après cette suppression.

— Scission : une entité est scindée. Les brins de l'entité appartiennent maintenant à plusieurs orbites du type de l'entité scindée. C'est le cas de la face de la figure 3.12b, scindée en 4 faces sur la figure 3.12c. Par exemple, si une entité est scindée au rejeu mais ne l'était pas au jeu initial, les entités scindées seront toutes appariées à l'entité qui était appariée à leur entité "mère".

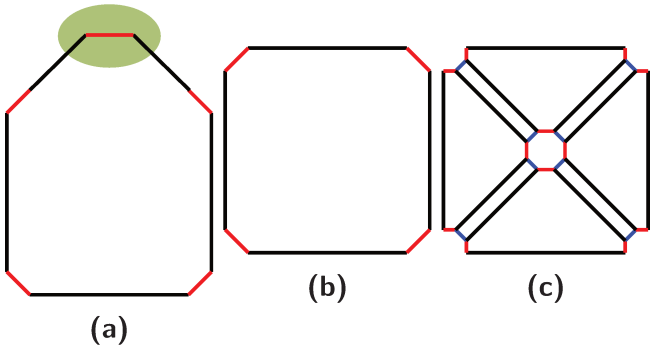


Figure 3.12 : Illustration des différents types de modification. a : Création d'un pentagone. b : Suppression du sommet supérieur de ce pentagone. c : Triangulation du carré résultant de cette suppression.

— Fusion : plusieurs entités fusionnent. L'entité est composée de brins appartenant précédemment à plusieurs entités de même type. Par exemple, les deux arêtes entourant le sommet vert sur la figure 3.12a, fusionnent en une arête sur la figure 3.12b. Par exemple, si une entité est fusionnée au rejeu mais ne l'était pas au jeu initial, l'entité fusionnée sera appariée aux entités qui était appariées à ses entités "mères".
— Modification : l'entité est simplement modifiée. Certains de ses brins ont été supprimés et/ou ajoutés par l'opération, mais l'entité n'a été ni scindée ni fusionnée. C'est le cas du carré de la figure 3.12b, qui est une modification du pentagone de la figure 3.12a.
— Aucune modification : l'entité n'est pas affectée par l'opération. C'est le cas par exemple des sommets non verts sur la figure 3.12a, qui n'ont pas été affectés par la suppression du sommet vert.

Le journal de bord sert à rendre compte du comportement de la règle, et est créé par l'utilisateur. Pour chaque type d'orbite, il indique les modifications qui permettent de passer des entités désignées par les nœuds de la partie gauche à celles désignées par ceux de la partie droite. Il indique également quel type d'orbite est nécessaire pour obtenir tous les brins permettant de créer l'entité par copie. Par exemple, pour la triangulation, le sommet central est composé de copies de tous les brins de la face initiale, il est donc créé à partir de l'orbite face.

La définition d'un journal de bord est la suivante :

Definition 9 (*Journaux de Bord*)

Soit n la dimension du modeleur de règles.
Soient `Split`, `Creat`, `Merge`, `No Eff.`, `Del.` *et* `Modif.` *les six types de modification pouvant impacter les différents types d'orbite lors de l'application d'une règle.*
Soit r une règle constituée de p nœuds dans sa partie gauche et q nœuds dans sa partie droite, avec p et $q \in \mathbb{N}$.
Le journal de bord de r est un tableau contenant 2^{n+1} entrées (une entrée par type d'orbite $\langle o \rangle$). Chaque entrée définit les modifications entraînées par la règle sur les orbites de type $\langle o \rangle$ et représentées par des graphes orientés acycliques contenant uniquement des chemins de longueur 1 et composés d'arcs :

- *reliant un nœud origine étiqueté par :*
  - *soit un sous-ensemble des nœuds de la partie gauche de r (noté $\{n_{g1}, ..., n_{gl}\}$ avec $1 \leq l \leq p$), un type d'orbite $\langle o' \rangle$, et filtrant l'ensemble des brins dont la réécriture forme l'orbite de type $\langle o \rangle$,*
  - *soit l'ensemble vide ;*
- *avec un nœud extrémité étiqueté par :*
  - *soit un sous-ensemble des nœuds de la partie droite de la règle r, noté $\{n_{d1}, ..., n_{dk}\}$, avec $1 \leq k \leq q$, et filtrant les brins formant une même orbite de type $\langle o \rangle$,*
  - *soit l'ensemble vide ;*
- *étiquetés par le label d'un type de modification, celui permettant de caractériser l'évolution de l'orbite associée au noeud d'origine.*

Pour construire $\langle o' \rangle$, on prend toutes les liaisons α existant dans un modèle de dimension n (α_0, α_1, et α_2 en dimension 2). On va ensuite isoler les brins copiés formant la nouvelle orbite, puis éliminer toutes les liaisons α qui ne permettent pas, à partir d'un des brins originaux, d'obtenir l'orbite cible $\langle o' \rangle$, c'est à dire les liaisons qui atteignent des brins qui ne sont ni à l'origine d'un brin copié faisant partie de l'orbite modifiée, ni des brins de l'orbite modifiée. Si on élimine tous les α, alors $\langle o' \rangle = \langle \rangle$.

Utilisons justement la règle de triangulation (rappelée sur la figure 3.13) comme exemple de la création du journal de bord. Nous allons étudier le cas des sommets (orbite $\langle 1, 2 \rangle$), en nous basant sur l'exemple de la figure 3.14, les quatre sommets résultant de la triangulation

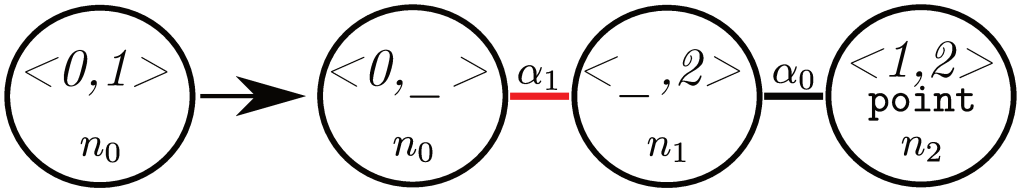


Figure 3.13 : Règle de triangulation

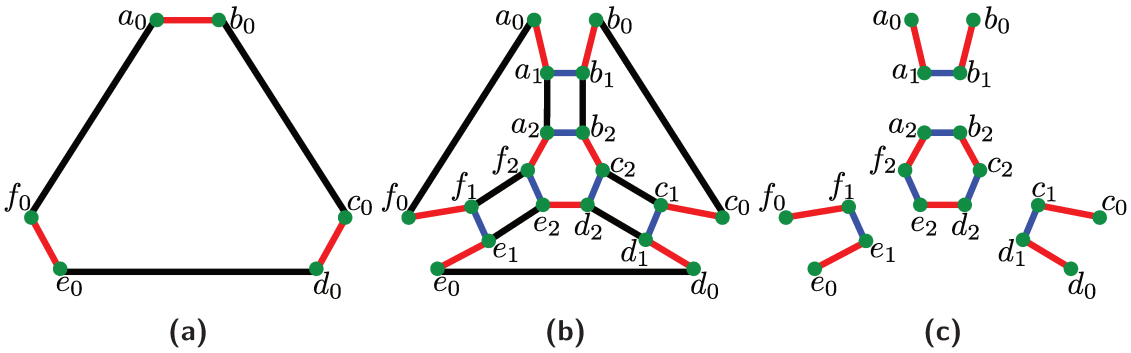


Figure 3.14 : Exemple d'application : Création d'un journal de bord. a : Triangle avant triangulation. b : Triangle après triangulation. c : Sommets de la face triangulée

.

étant isolés sur la figure 3.14c.

Commençons par le sommet central. Celui-ci est composé uniquement de brins filtrés par n_2. Ses brins, $a_2...f_2$ sont créés par copie des brins $a_0...f_0$ (c'est-à-dire, à partir de brins filtrés par n_0 dans la partie gauche), liés par le type d'orbite face $\langle 0, 1\rangle$. On élimine la liaison α_2 qui ne filtre aucun brin permettant de créer le sommet ou en faisant partie. C'est ce que l'on note $n_0.\langle 0, 1\rangle$. Enfin, les brins $a_2...f_2$ sont tous créés par l'opération, le sommet est donc créé. On crée donc un arc liant $n_0.\langle 0, 1\rangle$ à n_2 et portant la mention "creat." pour "creation". Le résultat est présenté sur la figure 3.15a.

On procède de la même manière pour les 3 autres sommets. Ceux-ci sont composés de brins filtrés par n_0 et n_1 dans la partie droite de la règle. Pour obtenir l'ensemble des brins composant un de ces sommets, on prend l'orbite $\langle 1, 2\rangle$ liée à un de ses brins filtrés par n_0 à gauche. Ainsi, les brins du sommet composé de a_0, a_1, b_1 et b_0 sont tous des copies de a_0 et b_0, liés par α_1 avant l'application de la règle. On considère également α_2, afin de tenir compte du fait que les sommets appartiennent également à toutes les faces qui leur sont incidentes. Au final, seule α_0 est éliminée, car elle permet d'atteindre des brins

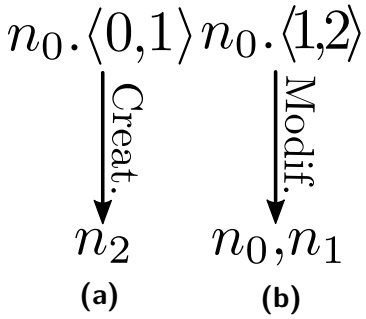


Figure 3.15 : Exemple d'application : Journal de bord de la triangulation - Sommets

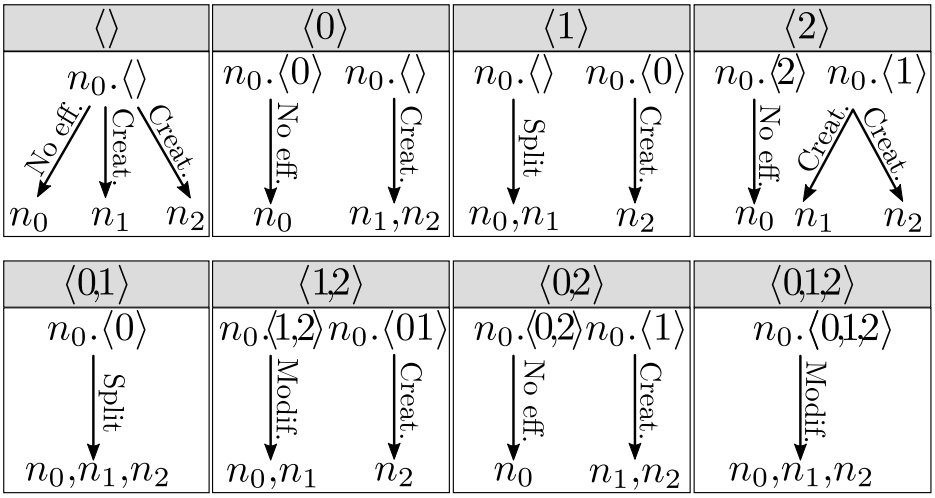


Figure 3.16 : Règle de triangulation - Journal de Bord

dont les copies forment différents sommets (a_0 et f_0 par exemple). Enfin, les sommets sont composés de brins créés (issus de n_1) et déjà existants (issus de n_0), ils sont donc modifiés. Par exemple, a_1 et b_1 sont créés mais a_0 et b_0 existent déjà. On crée donc un arc liant $n_0.\langle 1,2\rangle$ à n_0 et n_1 et portant la mention "Modif." pour "Modification". Le résultat est présenté sur la figure 3.15b.

Le journal de bord complet de la triangulation est présenté sur la figure 3.16. On y trouve le comportement attendu pour chaque type d'orbite. On constate que sur certaines orbites, $\langle\rangle$ par exemple, plusieurs arcs partent d'une même orbite attachée à un nœud. Cela est simplement dû au fait que, les trois flèches ayant la même origine, on simplifie le journal en ne mettant qu'une seule origine pour les trois, au lieu de les présenter côte à

côte. "No eff" signifie "No Effect", pas de modification. "Split" signifie scission.

L'intégralité des journaux de bord utilisés par notre exemple est donnée à l'annexe A.

En résumé, chaque règle possède un journal de bord, créé en même temps qu'elle dans l'éditeur de règles. Celui-ci permet de renseigner les modifications que la règle apporte aux entités. Nous verrons dans la section 4.2.1 comment il est possible d'utiliser le type de modification pour paramétrer le rejeu.

3.3 Journal d'historique

Grâce aux noms persistants, on connaît toutes les opérations ayant affecté une entité, et on peut identifier cette dernière de manière unique et non ambiguë au jeu initial. En outre, grâce aux journaux de bord, on connaît les modifications entraînées par une règle. Afin de définir précisément l'historique caractérisant l'entité, on crée un *journal d'historique* en combinant ces deux informations.

On construit les journaux d'historique d'une entité en remontant les différents identifiants de son nom persistant, de l'opération la plus récente à la plus ancienne. Un journal d'historique est ainsi lié à un identifiant persistant. Toutefois, quand un nom persistant ne compte qu'un seul brin, on peut parler de journal d'historique du nom PN pour parler du journal de son identifiant. Afin d'illustrer l'explication suivante, nous allons utiliser $PN_7 = \{\{1 - n_4; 2 - n_0; 3 - n_0; 6 - n_0\}\}\ .\langle 0, 1\rangle$, dont le journal d'historique est présenté sur la figure 3.17.

Pour cela, pour chaque identifiant persistant du nom (dans PN_7, il n'y en a qu'un, $\{1 - n_4; 2 - n_0; 3 - n_0; 6 - n_0\}$:

— On regarde d'abord sur quel type d'orbite TO_m s'applique l'opération, où m est le nombre d'éléments $[numéro - nœud]$ dans PI. Dans notre exemple, $m = 4$ et $TO_m = \langle 0, 1\rangle$ (flèche 1 sur la figure 3.17).

On récupère ensuite le journal de bord de cette opération, et, plus particulièrement, on sélectionne la partie concernant TO_m. Dans un deuxième temps, on récupère le dernier élément du nom persistant (PI_m), on étudie à quel nœud il fait référence (ici, il s'agit de $6-n_0$, et donc du nœud n_0), et dans l'extrait de journal de bord sélectionné, on cherche le graphe qui pointe vers ce nœud (ici, il s'agit de $n_0.\langle 0\rangle \rightarrow n_0, n_1, n_2$, via une scission). Cette étape est illustrée par la flèche 2 sur la figure 3.17.

Dans l'extrait de journal de bord, on récupérera ainsi deux informations : tout

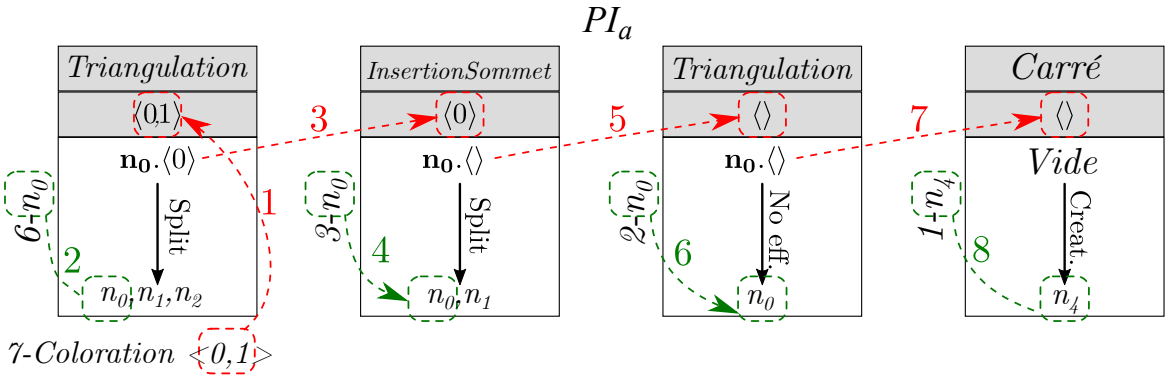


Figure 3.17 : Journal d'historique de PI_{7a}

d'abord, TO_{m-1}, le type d'orbite permettant de désigner les brins nécessaires à la construction de l'orbite de type TO_m liée à n_m, nœud de l'élément m (ici $TO_{m-1} = \langle 0\rangle$), et le type de modification ayant affecté l'orbite (ici, une scission).

— On s'intéresse dans un deuxième temps, et de manière similaire, à TO_{m-1}, et PI_{m-1}, le $(m-1)^{ème}$ élément de PI. Dans notre exemple, $TO_{m-1} = \langle 0\rangle$, et $PI_{m-1} = 3 - n_0$. Dans le journal de bord de l'opération correspondant à $m-1$, on regarde la partie concernant TO_{m-1}. Comme précédemment, on regarde quel nœud est impliqué dans PI_{m-1}, afin de récupérer l'information concernant le type de modification entraîné par la règle, et TO_{m-2}, le type d'orbite permettant de désigner les brins nécessaires à la construction de l'orbite de type TO_{m-1} liée à n_{m-1}.

Dans notre exemple, on étudie ainsi le journal de bord de l'insertion de sommet, et plus précisément l'extrait lié à $\langle 0\rangle$, notre TO_{m-1} (flèche 3). Dans cet extrait, on sélectionne le graphe qui pointe vers n_0, provenant de $PI_{m-1} = 3 - n_0$ (flèche 4). Celle-ci nous permet de savoir que le type de modification est une scission, et que $TO_{m-2} = \langle\rangle$.

— On répète cette deuxième étape en continuant de décrémenter $m-2$, jusqu'au premier élément de l'identifiant persistant, ce qui correspond à la figure 3.17 pour PN_7.

Note : Sur la figure 3.17, on utilise la notation PI_{7a}. Celle-ci signifie que l'on s'intéresse à l'identifiant persistant de a, utilisé par le nom PN_7. A la lecture, on sait ainsi immédiatement quel nom persistant est concerné. Cette notation est à ne pas confondre avec la notation PI_{a6}, qui permettait de savoir que la dernière étape à impacter a avant son utilisation était la 6. L'ordre lettre-numéro est inversé dans la nouvelle notation.

Notons que la première étape de la formation du journal d'historique est similaire aux suivantes, la seule différence étant que le type d'orbite est récupéré directement *via* la spécification paramétrique, et non *via* un journal de bord. Sur la figure 3.17, cela est mis en évidence par les différentes flèches.

La dernier étape étant celle de la création du brin, ou de son ancêtre le plus lointain, l'orbite TO_0 n'existe en réalité pas, car à cette étape, le brin (ou son ancêtre) est créé *ex nihilo*. On notera TO_0 : *vide*.

Un journal d'historique sera donc défini comme :

Definition 10 (*Journaux d'historique*)

Soit PN_E le nom persistant de l'entité E.
Soit PI_j l'un de ses identifiants persistants, et n le nombre d'éléments de PI_j.
Le journal d'historique attaché à PI_j est défini comme un tableau de n entrées. Soit l (avec $1 \leq l \leq n$) le numéro d'une entrée de ce tableau. Cette entrée est composée :

- *du $(n-l+1)^{\text{me}}$ élément $\{x - n_y\}$ de PI_j ;*
- *du type d'orbite $\langle o \rangle$ indiqué dans l'étiquette des nœuds origines du graphe extrait du journal de bord enregistré dans l'entrée $l-1$ du journal d'historique, ou celui de PN_E pour l'entrée 1 ;*
- *du graphe extrait du journal de bord de la règle utilisée par l'opération x, qui indique l'évolution des orbites de type $\langle o \rangle$ et contenant n_y dans l'étiquette d'un nœud extrémité.*

Ces entrées se suivent dans l'ordre inverse des éléments contenus dans PI_j, et donc dans l'ordre inverse des opérations ayant réécrit le brin correspondant à PI_j.

Un nom persistant est ainsi associé à autant de journaux d'historique qu'il est composé d'identifiants persistants.

Note : dans les exemples de journaux d'historique présentés, et afin de faciliter la lecture, on ajoute au dessus de chaque entrée le nom de l'opération x correspondante.

Un journal d'historique est ainsi une suite de graphes extraits de journaux de bord.

Nous allons maintenant détailler huit exemples, pour chaque nom persistant de la spécification de notre fil rouge à l'exception de PN_7 que nous venons de détailler. Pour rappel, les noms persistants sont présentés sur le tableau 3.1, et seront donnés à nouveau pour chaque paragraphe. Le code couleur utilisé dans la figure 3.17 sera également utilisé dans ces exemples, à savoir :

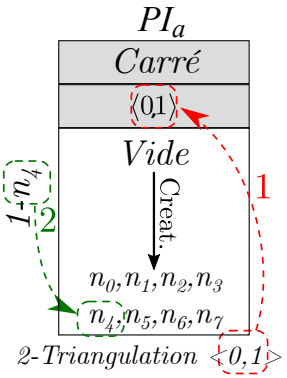


Figure 3.18 : Journal d'historique de PI_{1a}

— Des flèches rouges pour indiquer les liens entre les différents types d'orbite,
— Des flèches vertes pour indiquer les liens entre les nœuds indiqués par le nom persistant et les graphes des journaux de bord des règles.

Journal d'historique : PN_1

$PN_1 = \{\{1 - n_4\}\}.\langle 0, 1\rangle$. Il y a un seul identifiant persistant, et donc un seul journal d'historique, présenté sur la figure 3.18. L'opération qui fait appel à PN_1 est $2 - Triangulation$, effectuée sur l'orbite $\langle 0, 1\rangle$. Le brin de l'identifiant ayant précédemment été modifié par $1 - Carré$, on s'intéresse à son journal de bord, et plus précisément à son extrait concernant l'orbite $\langle 0, 1\rangle$, flèche 1, et dans celui-ci, au nœud n_4, flèche 2.

Il ne reste ensuite plus d'élément à remonter, le journal d'historique est terminé.

Journal d'historique : PN_2

$PN_2 = \{\{1 - n_3; 2 - n_0\}\}.\langle 0, 2\rangle$. Il y a un seul identifiant persistant, et donc un seul journal d'historique, présenté sur la figure 3.19. L'opération qui fait appel à PN_2 est $3 - InsertionSommet$, effectuée sur l'orbite $\langle 0, 2\rangle$ (une arête en 2D). Le brin de l'identifiant ayant précédemment été modifié par $2 - Triangulation$, on s'intéresse à son journal de bord [1], et plus précisément à son extrait concernant l'orbite $\langle 0, 2\rangle$, flèche 1, et dans celui-ci,

1. Comme indiqué précédemment, un journal de bord est associé à une règle ; et une opération faisant appel à une règle, on utilise ces deux termes de manière équivalente. Cependant, quand on parle du journal de bord de $2 - Triangulation$ par exemple, il s'agit bien du journal de bord de la règle de triangulation. Celui-ci sera donc le même que celui de $6 - Triangulation$, on ne crée pas un journal de bord à chaque

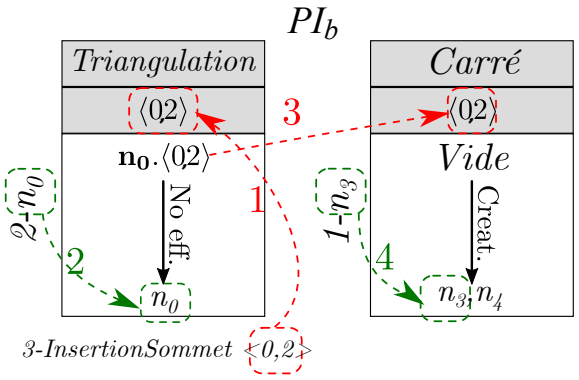


Figure 3.19 : Journal d'historique de PI_{2b}

au nœud n_0, flèche 2.

On continue ensuite à remonter l'identifiant persistant. L'extrait de journal de bord obtenu à l'étape précédente nous indique que l'orbite $\langle 0, 2\rangle$ liée à n_0 est construite à partir des brins d'une orbite de type $\langle 0, 2\rangle$ qui n'est pas modifiée par la règle de triangulation. Dans le journal de bord de $1 - Carré$, on prend donc l'extrait concernant ce dernier type d'orbite, flèche 3, et le nœud n_3, flèche 4.

Il ne reste ensuite plus d'élément à remonter, le journal d'historique est terminé.

Journal d'historique : PN_3

$PN_3 = \{\{1 - n_6; 2 - n_2\}\}.\langle 0, 2\rangle$. Le journal d'historique correspondant est présenté sur la figure 3.20. L'opération qui fait appel à PN_3 est $4 - InsertionSommet$, effectuée sur l'orbite $\langle 0, 2\rangle$ (flèche 1), et on s'intéresse au nœud n_2 pour l'opération $2 - Triangulation$ (flèche 2). Ensuite, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 2\rangle$ créée par la règle de triangulation, $\langle 1\rangle$ (flèche 3), et le graphe correspondant à n_6 pour l'opération $1 - Carré$ (flèche 4).

Journal d'historique : PN_4

$PN_4 = \{\{1 - n_5; 2 - n_2\}\}.\langle 0, 2\rangle$. Le journal d'historique correspondant est présenté sur la figure 3.21. L'opération qui fait appel à PN_4 est $5 - InsertionSommet$, effectuée sur

opération.

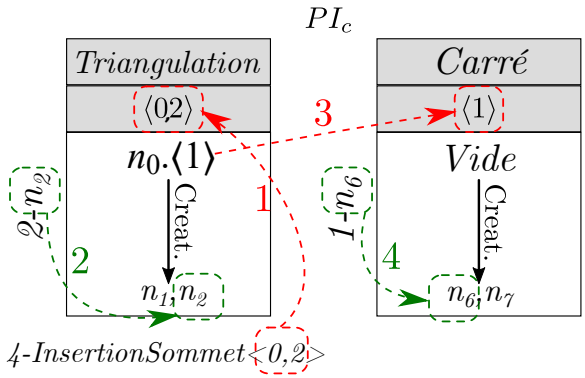


Figure 3.20 : Journal d'historique de PI_{3c}

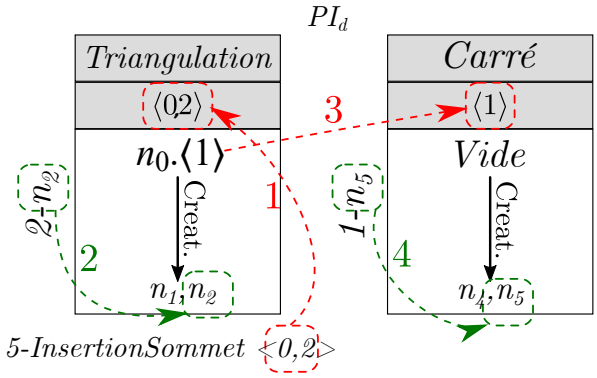


Figure 3.21 : Journal d'historique de PI_{4d}

l'orbite $\langle 0, 2\rangle$ (flèche 1), et on s'intéresse au nœud n_2 pour l'opération $2 - Triangulation$ (flèche 2). Ensuite, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 2\rangle$ créée par la règle de triangulation, $\langle 1\rangle$ (flèche 3), et le graphe correspondant à n_5 (flèche 4) pour l'opération $1 - Carré$.

Journaux d'historique : PN_5

$PN_5 = \{\{1 - n_3; 2 - n_0; 3 - n_0\}; \{1 - n_3; 2 - n_2\}; \{1 - n_4; 2 - n_1; 5 - n_1\}\}.\langle 0, 1\rangle$. Ce nom persistant est composé de trois identifiants persistants, et donc trois journaux d'historique, que nous allons présenter successivement.

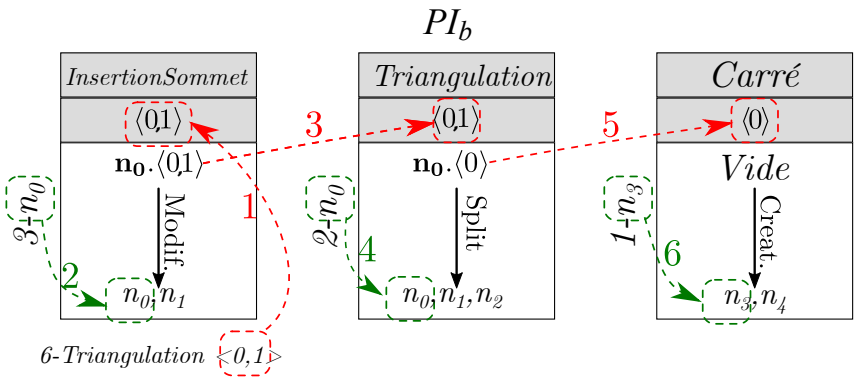


Figure 3.22 : Journal d'historique de PI_{5b}

PI_{b3}

Le journal d'historique correspondant est présenté sur la figure 3.22. L'opération qui fait appel à PN_5 est $6 - Triangulation$, effectuée sur l'orbite $\langle 0, 1\rangle$ (flèche 1), et on s'intéresse au nœud n_0 pour l'opération $3 - InsertionSommet$ (flèche 2). Ensuite, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de l'insertion de sommet, $\langle 0, 1\rangle$ (flèche 3), et le graphe correspondant à n_0 (flèche 4) pour l'opération $2 - Triangulation$. Enfin, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de triangulation, $\langle 0\rangle$ (flèche 5), et le graphe correspondant à n_3 (flèche 6) pour l'opération $1 - Carré$.

PI_{e2}

Le journal d'historique correspondant est présenté sur la figure 3.23. L'opération qui fait appel à PN_5 est $6 - Triangulation$, effectuée sur l'orbite $\langle 0, 1\rangle$ (flèche 1), et on s'intéresse au nœud n_2 pour l'opération $2 - Triangulation$ (flèche 2). Ensuite, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de triangulation, $\langle 0\rangle$ (flèche 3), et le graphe correspondant à n_3 (flèche 4) pour l'opération $1 - Carré$.

PI_{f5}

Le journal d'historique correspondant est présenté sur la figure 3.24. L'opération qui fait appel à PN_5 est $6 - Triangulation$, effectuée sur l'orbite $\langle 0, 1\rangle$ (flèche 1), et on s'intéresse au nœud n_1 pour l'opération $5 - InsertionSommet$ (flèche 2). Ensuite, on récupère l'orbite

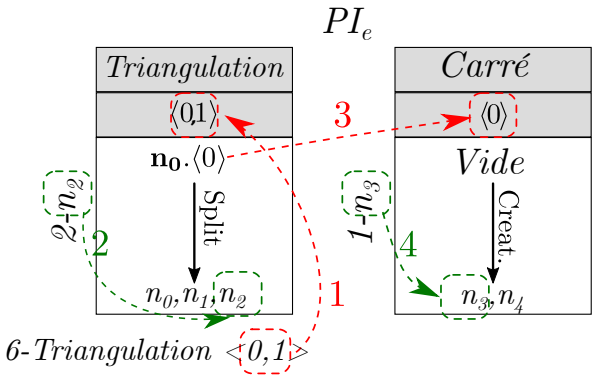


Figure 3.23 : Journal d'historique de PI_{5e}

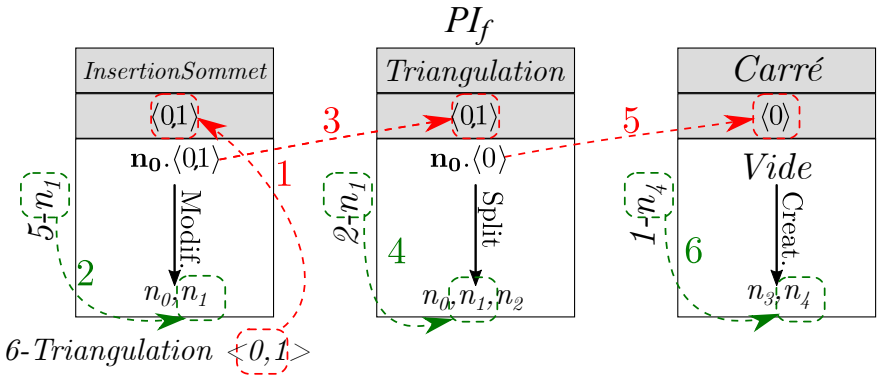


Figure 3.24 : Journal d'historique de PI_{5f}

à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de l'insertion de sommet, $\langle 0, 1\rangle$ (flèche 3), et le graphe correspondant à n_1 (flèche 4) pour l'opération $2 - Triangulation$. Enfin, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de la triangulation, $\langle 0\rangle$ (flèche 5), et le graphe correspondant à n_4 (flèche 6) pour l'opération $1 - Carré$.

Journal d'historique : PN_6

Le journal d'historique correspondant est présenté sur la figure 3.25. L'opération qui fait appel à PN_6 est $7 - Coloration$, effectuée sur l'orbite $\langle 0, 1\rangle$ (flèche 1), et on s'intéresse au nœud n_0 pour l'opération $6 - Triangulation$ (flèche 2). Ensuite, on récupère l'orbite à l'origine de l'orbite de type $\langle 0, 1\rangle$ créée par la règle de la triangulation, $\langle 0\rangle$ (flèche 3), et le graphe correspondant à n_0 (flèche 4) pour l'opération $5 - InsertionSommet$. Dans un troisième temps, on récupère l'orbite à l'origine de l'orbite de type $\langle 0\rangle$ créée par la règle

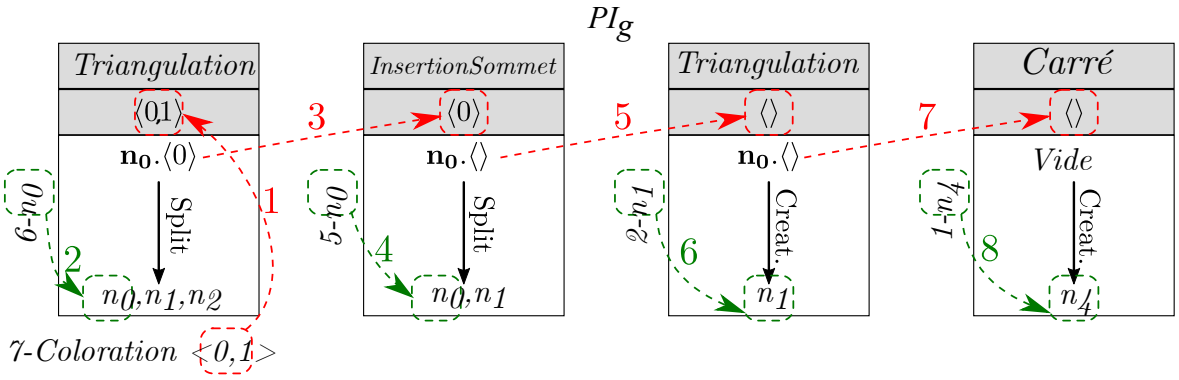


Figure 3.25 : Journal d'historique de PI_{6g}

de l'insertion de sommet, $\langle\rangle$ (flèche 5), et le graphe correspondant à n_1 (flèche 6) pour l'opération $2 - Triangulation$. Enfin, on récupère l'orbite à l'origine de l'orbite de type $\langle\rangle$ créée par la règle de triangulation, $\langle\rangle$ (flèche 7), et le graphe correspondant à n_4 (flèche 8) pour l'opération $1 - Carré$.

Pour une consultation plus rapide en cours de lecture, tous ces journaux d'historique sont regroupés dans l'annexe B, section B.1.

3.4 Résumé

En résumé, nous avons défini trois principaux outils :

— Le *Nom Persistant*, créé grâce aux *Identifiants Persistants*. Le nom rend compte de toutes les opérations ayant affecté l'entité.

— Le *Journal de Bord* des règles. Il est attaché à une règle, et pourrait potentiellement être utilisé dans un contexte autre que celui de cette thèse, pour d'autres utilisations des règles. Il permet de connaître, via l'application d'une règle dans un contexte donné, la manière dont évolueront les entités, information manquante dans le Nom Persistant.

— Le *Journal d'Historique* des entités. Celui-ci va mettre en relation les Noms Persistants et les Journaux de Bord afin de permettre de suivre l'évolution d'une entité. De plus, afin de ne jamais avoir à modifier le Nom Persistant, c'est le Journal d'Historique qui portera la trace des modifications de l'entité au rejeu.

Notons que tous ces outils sont élaborés avant de commencer à rejouer les opérations. Les

Journaux de Bord sont construits dès la création des règles, tandis que les Noms Persistants et les Journaux d'Historique le sont pendant le jeu initial, au moment où une entité est désignée pour être utilisée comme paramètre d'une opération, mais avant de commencer à rejouer les opérations.

Show must go on

Queen

4

Rejeu à base de règles de transformation de graphe

Sommaire

Maintenant que tous nos outils sont établis, nous allons voir dans ce chapitre comment les utiliser pour apparier les entités et réaliser le rejeu, tout en continuant de nous appuyer sur notre exemple fil rouge afin d'illustrer notre méthode.

La figure 4.1 rappelle l'exemple fil rouge introduit précédemment. La figure 4.2 présente le rejeu de ce premier modèle. Notons que sur cette figure n'apparaissent les noms d'aucun brin. En effet, les entités sont récupérées directement par appariement, on n'a donc besoin que du nom persistant. Cependant, en cas d'opération ajoutée, l'utilisateur doit renseigner un brin d'application, comme pour une opération initiale.

Au jeu initial, nous avions la spécification :

— $1 - Carré()$
— $2 - Triangulation(PN_1)$
— $3 - InsertionSommet(PN_2)$
— $4 - InsertionSommet(PN_3)$
— $5 - InsertionSommet(PN_4)$
— $6 - Triangulation(PN_5)$
— $7 - Coloration(PN_6, Bleu, PN_7, Vert)$

et au rejeu, nous avons désormais :

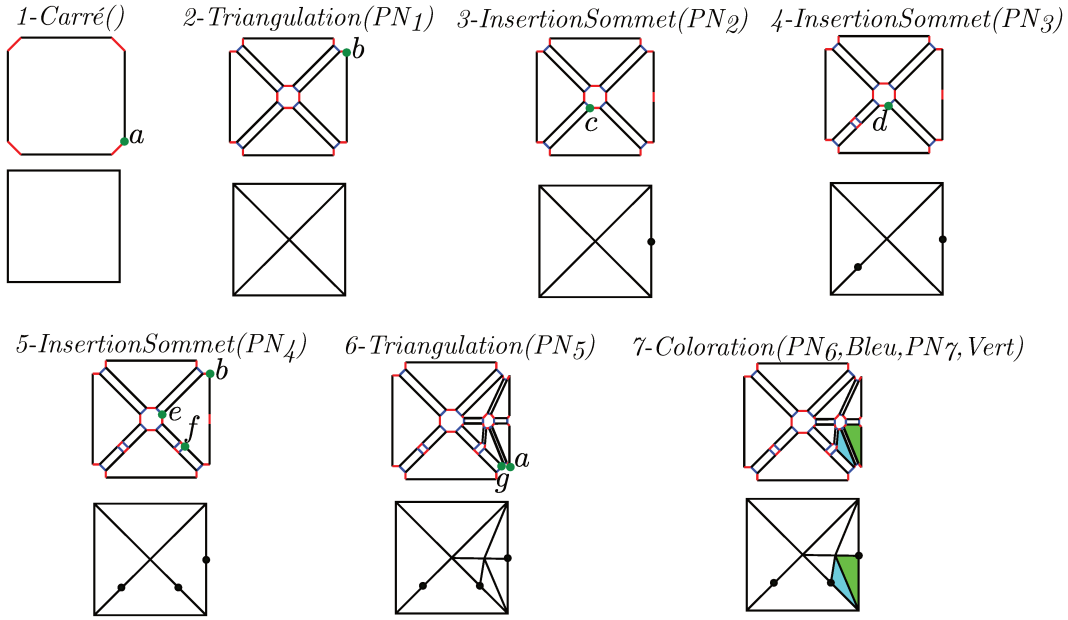


Figure 4.1 : Exemple "Fil Rouge" permettant de dérouler notre mécanisme de rejeu

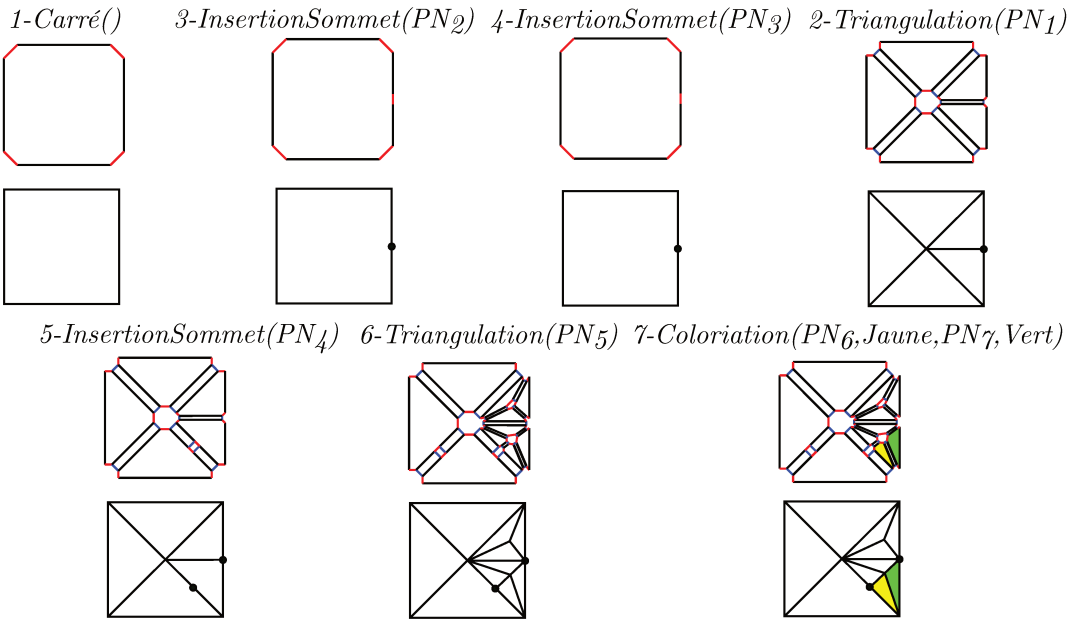


Figure 4.2 : Exemple "Fil Rouge" rejoué

— $1 - Carré()$
— $3 - InsertionSommet(PN_2)$
— $4 - InsertionSommet(PN_3)$
— $2 - Triangulation(PN_1)$
— $5 - InsertionSommet(PN_4)$
— $6 - Triangulation(PN_5)$
— $7 - Coloration(PN_6, Jaune, PN_7, Vert)$

Notons que dans cet exemple, nous avons changé l'ordre des opérations au rejeu, $2 - Triangulation$ ayant lieu plus tard qu'initialement, et nous avons modifié un paramètre, "Bleu" étant devenu "Jaune". Les autres opérations n'ont subi aucune modification.

Nous allons maintenant étudier les cinq différents *types d'édition* d'opération existants, à savoir les trois présentés ci-dessus (aucun changement, changement de paramètre et changement d'ordre) ainsi que l'ajout et la suppression d'opération (dont nous verrons qu'ils font partie des mécanismes permettant de déplacer une opération, et sont donc en réalité déjà présents dans notre exemple fil rouge). Ensuite, nous détaillerons le mécanisme des arbres d'appariement, permettant de lier entités initiales et entités rejouées. Enfin, nous verrons comme réaliser le rejeu de notre spécification.

4.1 Cinq types d'édition d'opérations

Nous allons donc commencer par détailler les cinq types d'édition impactant les opérations, et étudier la manière dont ils affectent les journaux d'historique des entités le cas échéant. En effet, afin de réaliser l'appariement des entités, leur journaux d'historique peuvent être mis à jour pendant le déroulement du rejeu.

4.1.1 Aucun changement d'ordre ou de paramètre / changement de paramètre(s) géométrique(s)

Ce sont les deux types d'édition les plus simples. L'opération reste telle qu'elle était au jeu initial. Au niveau du journal d'historique, il n'y aura pas de changement.

À noter cependant : le changement de paramètre ne peut pas concerner un des noms persistants (paramètres topologiques). En effet, prenons par exemple $3 - InsertionSommet(PN_2)$. Si l'on change PN_2, particulièrement si l'on sélectionne une autre orbite, nous n'avons plus

la même opération. Dans ce cas, on aurait une suppression de $3-InsertionSommet(PN_2)$, et l'ajout d'une nouvelle opération à sa place.

Dans notre exemple fil rouge, les opérations 1, 3, 4, 5, et 6 ne subissent aucun changement. L'opération 7, elle, subit un changement de paramètre au niveau d'une des couleurs.

4.1.2 Ajout

Dans ce cas, une opération est ajoutée à un endroit quelconque de la spécification (figure 4.3a). À première vue, il n'y en a pas dans notre fil rouge. Cependant, si l'on se place du point de vue de $2-Triangulation$, les opérations 3 et 4 sont considérées comme ajoutées au rejeu, comme présenté sur les figures 4.3c et 4.3d. On y voit que pour les opérations déplacées après leur position originale (notre cas, figure 4.3c), les opérations intermédiaires (3 et 4) peuvent être considérées comme ajoutées. Dans le cas inverse, celui des opérations déplacées vers le haut, les opérations intermédiaires voient l'opération déplacée comme ajoutée.

Au niveau des journaux d'historique, il n'y a pas de modification. En effet, on ne peut *a priori* pas savoir si l'ajout affecte ou non une entité. Par exemple, la figure 4.4 nous montre un jeu initial puis deux rejeux qui pourraient en découler. Sur le premier, la face f_2 qui sert de support à l'opération x n'est pas affectée par le rejeu, alors qu'elle l'est sur le deuxième. On ne peut donc pas inscrire l'information dans le journal d'historique si elle n'affecte pas l'entité. Nous verrons cependant à la section 4.2 comment l'information est prise en compte.

Note : dans le cas d'opérations réellement ajoutées, on décide, par convention, de faire repartir l'identifiant de la première opération à 1, et d'ajouter le préfixe "A" devant ce dernier. Si on ajoute par exemple une insertion d'arête entre deux sommets entre l'opération 5 et la 6, celle-ci sera nommée $A1-InsertionArête(s_1, s_2)$ et une deuxième opération, ajoutée après $7-Coloration$, portera le nom $A2-Operation$.

4.1.3 Suppression

Dans ce cas-là, une des opérations de la spécification initiale est supprimée (figure 4.3b). Comme pour l'ajout, à première vue, il n'y en a pas dans notre spécification. Cependant, du point de vue des opérations 3 et 4, l'opération $2-Triangulation$ est considérée comme

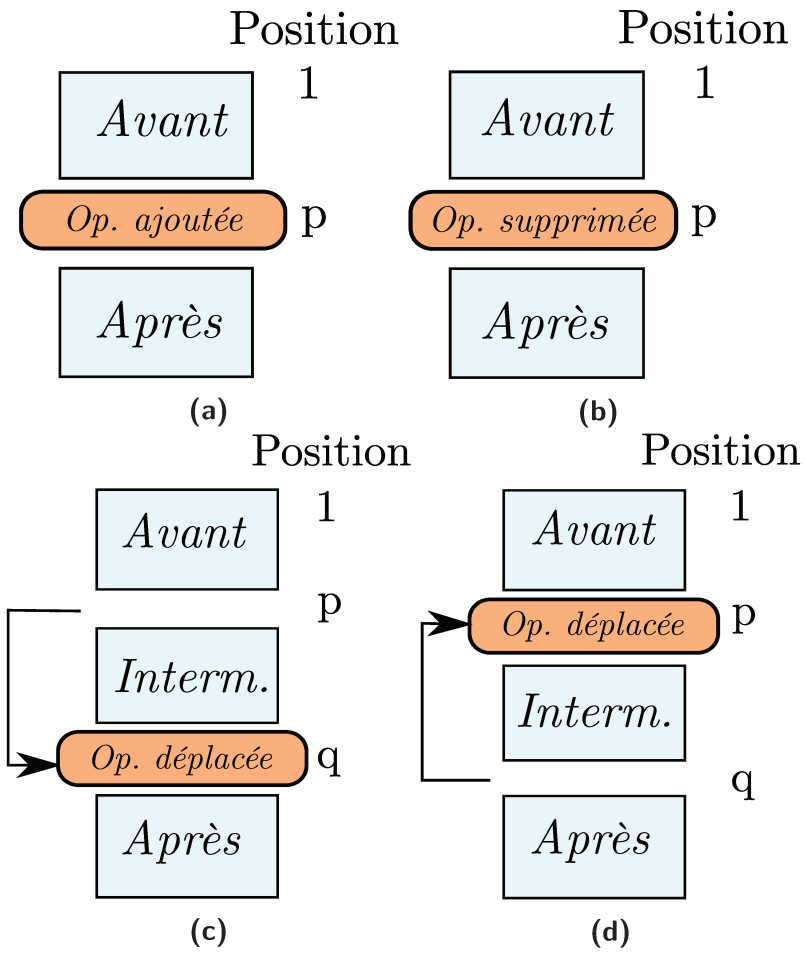


Figure 4.3 : a : Ajout d'opération. b : Suppression d'opération. c : Déplacement d'opération vers le bas (Interm. signifie Intermédiaires). d : Déplacement d'opération vers le haut

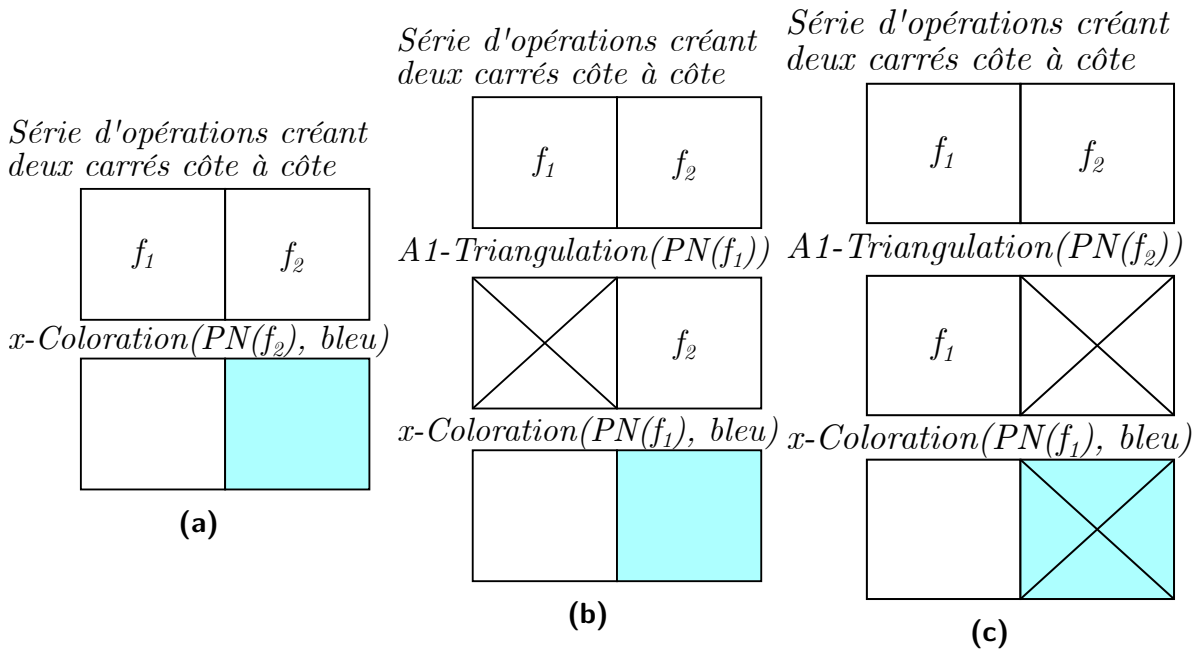


Figure 4.4 : Impact d'un ajout sur une entité. a : Modèle initial (x est le numéro de l'opération Coloration). b : Rejeu en ajoutant une triangulation sur f_1, pas d'impact sur f_2. c : Rejeu en ajoutant une triangulation sur f_2, la face est subdivisée et toutes ses filles sont colorées.

supprimée, car elle a maintenant lieu après. C'est ce que l'on constate sur la figure 4.3c pour les opérations appartenant au bloc intermédiaire. À l'inverse, une opération déplacée en amont de sa position d'origine perçoit les opérations intermédiaires comme supprimées (figure 4.3d).

Au niveau du journal d'historique, supprimer une opération revient à la "sauter". Cependant, cela ne signifie pas que l'on n'en tiendra pas compte. En effet, il faut retenir quelle était l'orbite qui caractérisait l'entité après l'opération supprimée. Pour PI_{2b} et PI_{3c} dans notre exemple (figures 4.5 et 4.6), on ne tiendra ainsi pas compte de la triangulation, mais on retiendra qu'après celle-ci, on s'intéresse à l'orbite $\langle 0, 2\rangle$. En effet, si on ne le fait pas, on obtiendra une inadéquation entre le type d'orbite sur lequel s'applique l'insertion d'arête, $\langle 0, 2\rangle$, et le type d'orbite transmis par le journal de PI_{3c}, $\langle 1\rangle$. Concrètement, cela revient à ignorer la flèche désignant le nœud de l'étape, aboutissant par exemple pour PI_{2b} et PI_{3c} aux journaux d'historique présentés sur la figure 4.7, avec une suppression de la flèche

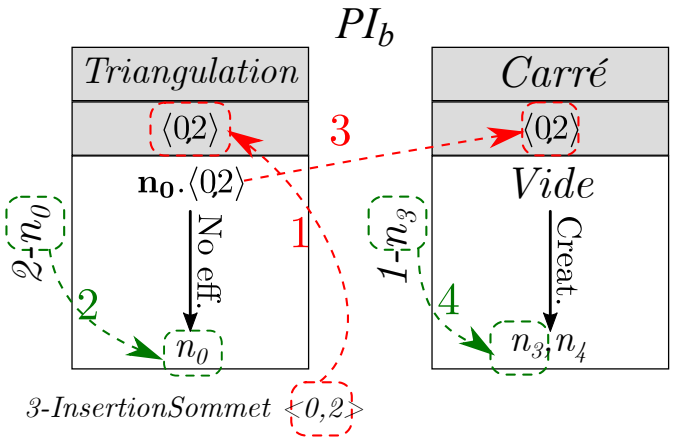


Figure 4.5 : Journal d'historique de PI_{2b}

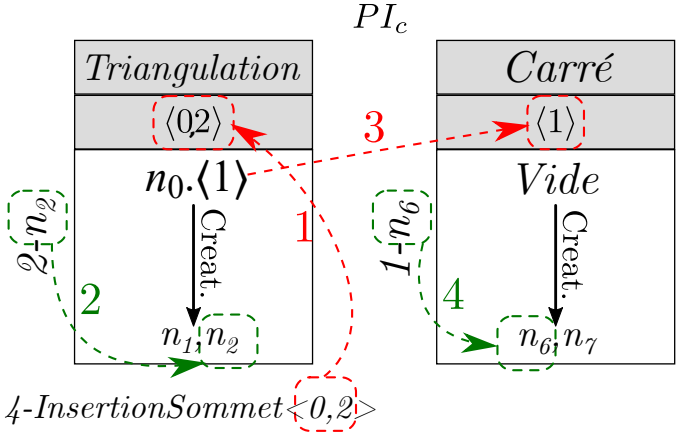


Figure 4.6 : Journal d'historique de PI_{3c}

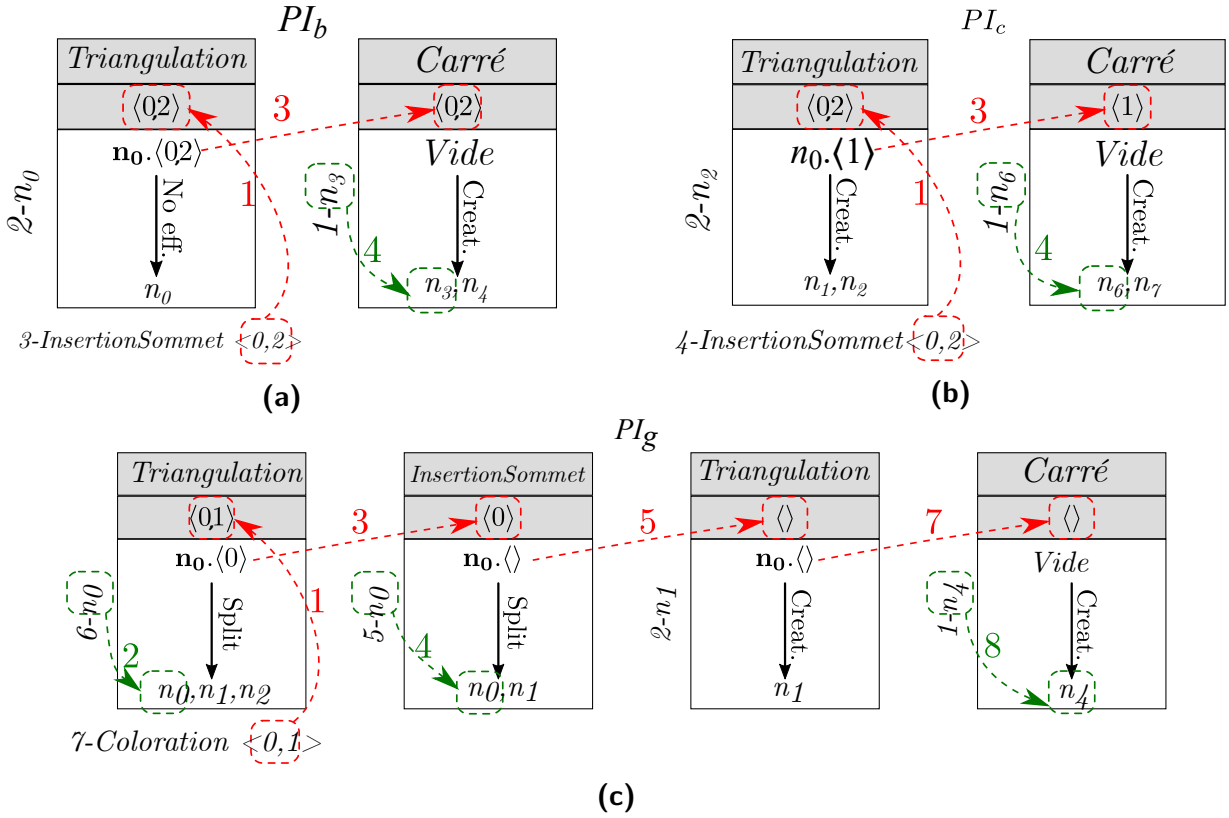


Figure 4.7 : Modification de journaux d'historique par la suppression. a : PI_{2b}. b : PI_{3c}. c : suppression hypothétique de $2 - Triangulation$ et impact sur PI_{6g}

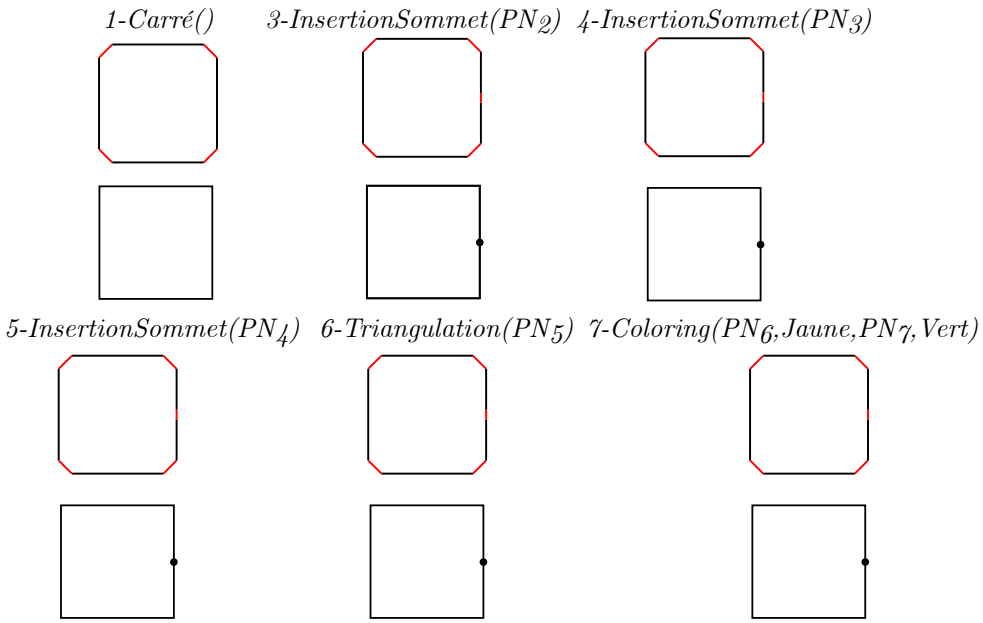


Figure 4.8 : Suppression hypothétique de $2 - Triangulation$

numéro 2 sur les deux journaux, par rapport aux journaux originaux.

Bien que dans notre exemple fil rouge cette "suppression" n'affecte pas PI_{6g}, qui est le paramètre d'une opération située dans le bloc "Après" et non pas dans le bloc "Interm.", la figure 4.7c présente tout de même le résultat de la suppression hypothétique de $2 - Triangulation$, afin de donner un exemple de suppression n'affectant pas la dernière étape d'un journal d'historique. La figure 4.8 présente le résultat que nous aurions obtenu si cette suppression avait été réalisée. À cause du paramétrage (section 4.2.1) et / ou du fait que supprimer l'opération 2 entraîne la non-application d'autres opérations (section 4.2.2), les opérations 4, 5, 6 et 7 n'auraient pas pu être rejouées.

Note : la suppression d'une opération ne signifie pas qu'on arrête d'apparier les entités qu'elle affectait. Nous verrons cela plus en détail dans la section 4.2.1, qui concerne le paramétrage du rejeu.

4.1.4 Changement d'ordre

Enfin, dans ce dernier cas, une opération est déplacée, soit en amont (figure 4.3d), soit en aval (figure 4.3c) de sa position dans la spécification. Dans notre exemple, c'est le cas de l'opération $2 - Triangulation$. Un changement d'ordre a un impact sur toutes les opérations

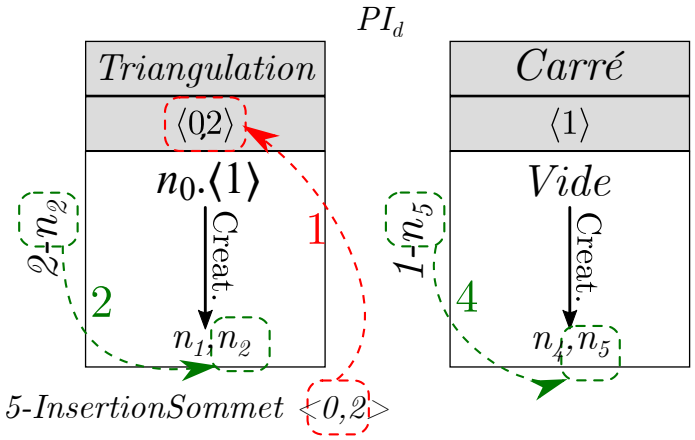


Figure 4.9 : Journal d'historique modifié de PI_{4d}

situées dans le bloc "après" sur les figures 4.3c et 4.3d, les opérations "Interm." considérant, comme indiqué précédemment, l'opération déplacée comme ajoutée si déplacée en amont, supprimée si déplacée en aval.

Au niveau des journaux d'historique des noms persistants des opérations "Après", les opérations sont réorganisées : il faut donc recalculer ces journaux d'historique pour tenir compte du nouvel ordre. Pour cela, on recalcule tous les éléments du journal liés aux opérations "Interm." et à l'opération déplacée, de la même manière que le journal avait précédemment été calculé (section 3.3). Les parties correspondant aux opérations "Avant" et "Après", elles, ne sont pas recalculées, car précédant le changement d'ordre pour les premières, et non affectées par le changement d'ordre pour les dernières, le journal d'historique étant construit dans le sens inverse des opérations. Dans notre exemple, nous devons donc recalculer les journaux des identifiants de PN_4, PN_5, PN_6 et PN_7. Étudions-les un par un.

Journal d'historique recalculé : PN_4

$PN_4 = \{\{1 - n_5; 2 - n_2\}\}.\langle 0, 2\rangle$.

Ici, pas d'impact des opérations "Interm." 3 et 4, car les opérations 3 et 4 n'apparaissent pas dans PN_4. Le fait que l'opération $2 - Triangulation$ soit déplacée après ces dernières n'a donc pas d'impact direct sur PI_{d2}. Le journal d'historique de PI_{4d} reste donc inchangé (figure 4.9).

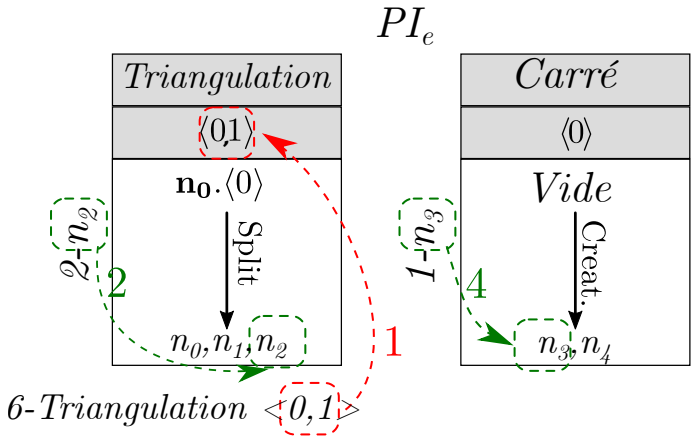


Figure 4.10 : Journal d'historique modifié de PI_{5e}

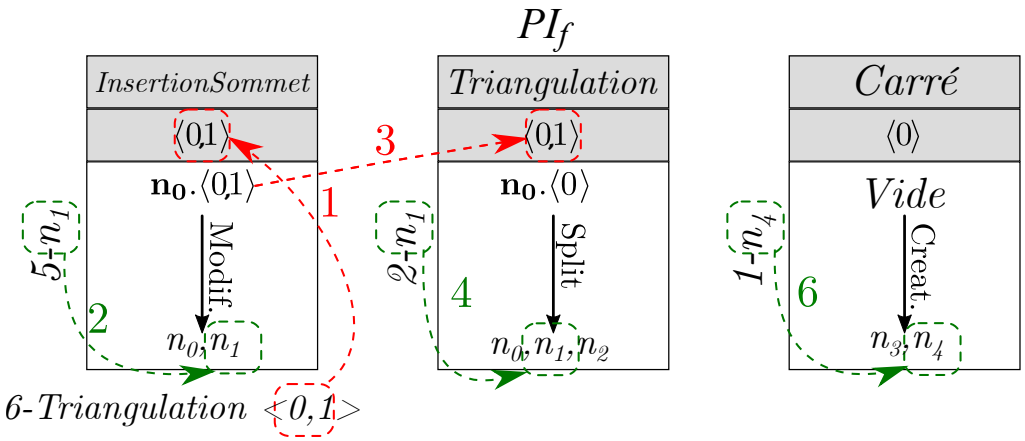


Figure 4.11 : Journal d'historique modifié de PI_{5f}

Journal d'historique recalculé : PN_5

$PN_5 = \{\{1-n_3; 2-n_0; 3-n_0\}; \{1-n_3; 2-n_2\}; \{1-n_4; 2-n_1; 5-n_1\}\}.\langle 0,1\rangle$.

Ici, deux cas. Tout d'abord, les deux derniers identifiants persistants PI_{e2} et PI_{f5}, ne sont pas impactés par le changement d'ordre, leurs journaux d'historique restent donc inchangés (figure 4.10 et figure 4.11).

PI_{b3} est en revanche affecté, et devient $PI'_{b3} = \{1-n_3; 3-n_0; 2-n_0\}$. On recalcule donc son journal d'historique, ce qui donne le résultat affiché dans la figure 4.12.

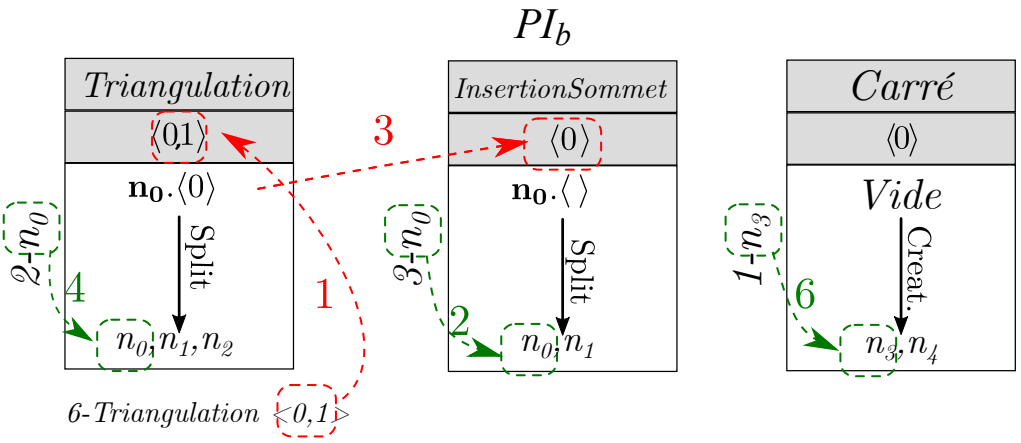


Figure 4.12 : Journal d'historique modifié de PI_{5b}

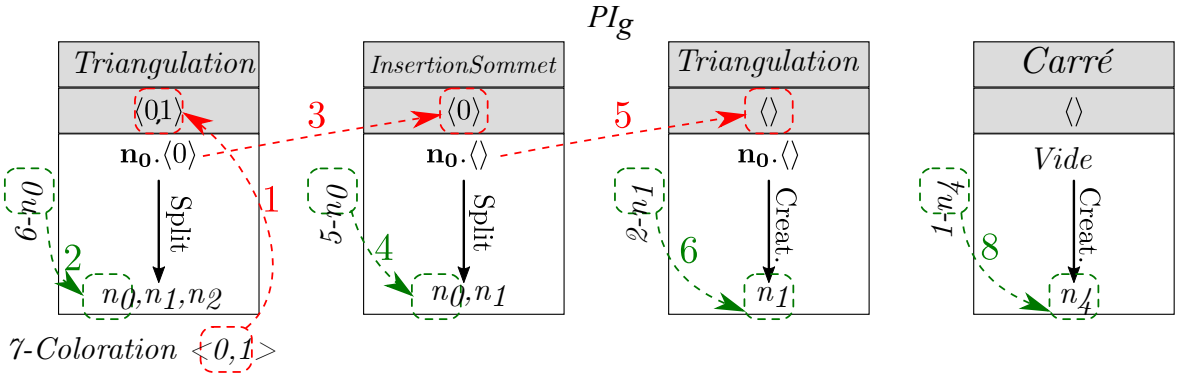


Figure 4.13 : Journal d'historique modifié de PI_{6g}

Journal d'historique recalculé : PN_6

$PN_6 = \{\{1 - n_4; 2 - n_1; 5 - n_0; 6 - n_0\}\}\,.\,\langle 0, 1\rangle$.

A nouveau, il n'y a pas d'impact direct du changement d'ordre, le journal reste identique (figure 4.13).

Journal d'historique recalculé : PN_7

$PN_7 = \{\{1 - n_4; 2 - n_0; 3 - n_0; 6 - n_0\}\}\,.\langle 0, 1\rangle$.

Enfin, dans ce dernier cas, le nom est affecté par le changement d'ordre, car PI_{a6} devient $PI'_{a6} = \{1 - n_4; 3 - n_0; 2 - n_0; 6 - n_0\}$. Le journal d'historique est donc recalculé, et présenté sur la figure 4.14.

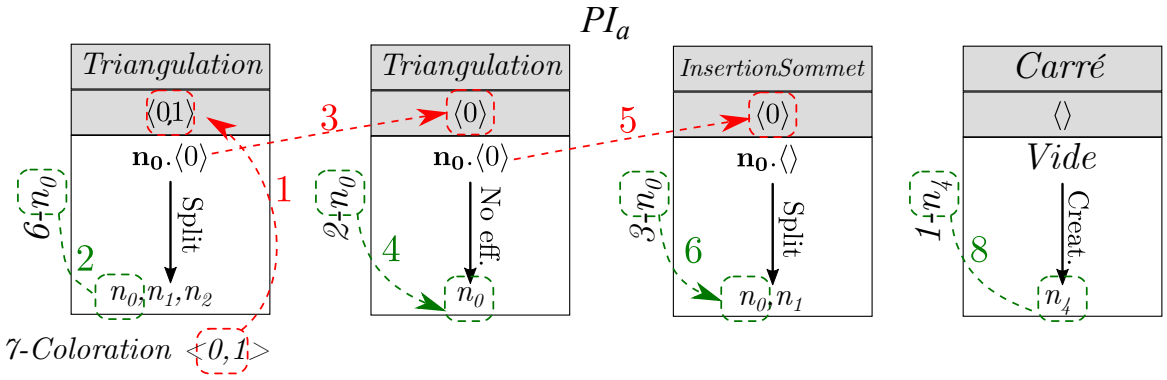


Figure 4.14 : Journal d'historique modifié de PI_{7a}

Comme précédemment, les nouveaux journaux d'historique sont présentés en annexe, dans la section B.2.

4.2 Arbres d'appariement

Nous arrivons maintenant au dernier élément de notre algorithme : les arbres d'appariement. Ce sont eux qui permettent d'apparier les entités du jeu initial et du rejeu, mais également de paramétrer cet appariement.

Les feuilles des arbres d'appariement sont des orbites, désignées par un ensemble *brin.type.d'orbite* et leurs branches sont un couple formé par les types de modifications qui affectent les orbites, et les types d'édition qui affectent les opérations. Chaque arbre a pour racine l'un des identifiants persistants d'un nom, et suivra les orbites appariées avec cet identifiant. Quand on n'a plus besoin d'un arbre, c'est-à-dire quand l'opération qui utilise son nom racine a été appliquée, on supprime l'arbre.

Les arbres d'appariement seront définis de la manière suivante :

Definition 11 (*Arbre d'appariement*)

Soit $PN_E = \{PI_1...PI_q\}.\langle o \rangle$ le nom persistant d'une entité E constitué de q identifiants persistants.
Soient `Rejeu Identique`, `Rejeu avec changement de Paramètre`, `Ajout`, `Suppression` *et* `Changement d'ordre` *cinq types d'édition.*
L'arbre d'appariement de PN_E est défini comme un arbre constitué :

- *d'une racine : PN_E ;*
- *de nœuds dont :*
  - *les premiers (les fils de la racine) correspondent aux différents PI_j, avec $1 \leq j \leq q$.*
  - *les suivants correspondent aux différentes orbites indiquées dans les journaux d'historique attachés à chaque PI_j ;*
- *de branches étiquetées par le type d'édition et le type de modification impactant les nœuds.*

Afin de construire ces arbres et de les mettre à jour tout au long du rejeu, nous avons besoin de l'ensemble des concepts vu jusqu'ici, et principalement des journaux d'historique (sections 3.3 et 4.1), et donc des noms persistants (section 3.1), des journaux de bord, des types de modification (section 3.2) et des types d'édition (section 4.1).

Nous avons vu dans la section précédente cinq types d'édition. Chaque type d'édition donne lieu à une de mise à jour / création des arbres d'appariement qui peuvent être regroupées en trois cas distincts de mises à jour / création. Le premier concerne les types d'édition opérations supprimées, avec des paramètres géométriques modifiés, ou sans changement, le deuxième concerne les opérations déplacées, et le troisième, les opérations ajoutées. En effet, pour ces dernières opérations, nous avons vu à la section précédente qu'il n'était pas possible de mettre à jour le journal d'historique des noms. C'est donc grâce aux arbres que l'on pourra prendre en compte les changements qu'elles entraînent. Nous allons donc tout d'abord expliquer comment paramétrer le rejeu grâce à ces arbres, puis étudier ces trois cas de mise à jour, et enfin, nous appliquerons cela à notre exemple fil rouge.

4.2.1 Paramétrage du rejeu

Étudions tout d'abord comment paramétrer le rejeu grâce aux arbres d'appariement.

Nous avons 5 types d'édition : Rejeu à l'identique, Changement de paramètres géométriques, Suppression, Ajout et Changement d'ordre, et 6 types de modifications : Création, Suppression, Scission, Fusion, Modification, Aucune modification, pour un total de 25 combinaisons, le changement d'ordre ne s'associant à aucun type de modification. Une branche portera ainsi le label "Aj. Creat." pour une création ajoutée, "Sup. Split" pour une scission supprimée, "Dépl." pour signaler le déplacement d'une opération, "Chg. P. No Eff." pour un changement de paramètre sur une opération sans effet, et enfin, "Mer." uniquement pour une fusion résultant d'une opération n'ayant subi aucun changement (afin de rendre la lecture plus facile à l'œil humain). Afin de mieux distinguer ces deux types, on utilisera ainsi le français pour les types de d'édition, et l'anglais pour les types de modification.

C'est donc en fonction de ces associations que l'on paramètre le rejeu. Par exemple, dans notre cas, nous avons décidé qu'en cas de "Sup. Creat.", c'est-à-dire de la suppression d'une opération entraînant une création, on décide de ne plus suivre la branche, ni aucune des branches du nom persistant qui lui est liée, car l'entité n'étant pas créée, elle n'existe pas, et ne peut pas avoir d'appariement. De même, on ne suit plus la branche en cas de "Sup. Split", les différentes entités créées par une scission n'existant plus. À l'inverse, pour les autres associations, on continue le suivi. L'utilisateur peut donc entièrement paramétrer le rejeu en indiquant pour quelles associations il réalise le suivi, et pour lesquelles il ne le fait pas. Il peut également indiquer s'il souhaite étendre l'arrêt de suivi à tous les arbres d'un nom, ou seulement à l'arbre concerné.

De plus, ce système est suffisamment souple pour permettre également la mise en place d'un paramétrage spécifique par opération, le paramétrage étant le même qu'expliqué précédemment, mais ne s'appliquant qu'à une opération donnée. Par exemple, on pourrait ne pas prendre en compte les branches "Dépl." découlant l'opération $2 - Triangulation$. Aucune des opérations 5, 6 et 7 ne seraient alors rejouées, toutes étant affectées par le changement d'ordre. De même, on pourrait étendre un paramétrage à une règle. Dans ce cas, la branche "Dépl." ne sera prise en compte ni pour $2 - Triangulation$ ni pour $6 - Triangulation$.

4.2.2 Opérations supprimées, modifiées ou sans changement

Dans le cas de ces opérations, on suit simplement les journaux d'historique des différents noms pour construire ou mettre à jour les arbres. Pour cela, on procède en deux étapes,

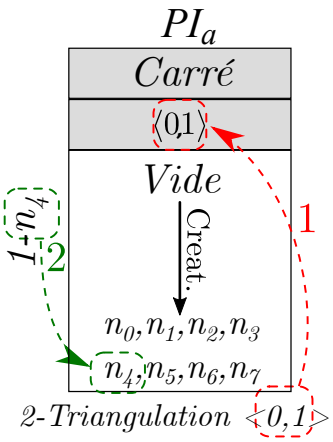


Figure 4.15 : Journal d'historique de PI_{1a}

que l'on va détailler sur la première opération de notre fil rouge, $1 - Carré$, et sur PN_1 :

— Dans un premier temps, on sélectionne le brin qui servira de support à l'orbite. Pour cela, on sélectionne le brin correspondant à la copie indiquée dans l'extrait de journal d'historique lié à la règle. Prenons l'exemple de PN_1 (figure 4.15). On sait qu'il faut prendre le brin lié à n_4 créé *ex nihilo*, ce qui correspond au brin a' sur la figure 4.16a.

— Dans un deuxième temps, on relie ce brin au brin précédent dans l'arbre concerné, en étiquetant la branche par l'association du type d'édition affectant l'opération (sauf dans le cas d'édition "sans modification", comme indiqué en section 4.2.1), et le type de modification indiquée dans le journal d'historique. Pour notre exemple, ce sera donc "Creat.", l'opération étant rejouée sans modification. On obtient ainsi l'arbre présenté sur la figure 4.16b.

Dans le cas de la suppression d'opération, on se contente de créer la nouvelle feuille en récupérant le brin indiqué par l'ancienne, auquel on associe le type d'orbite indiqué dans son journal d'historique. Par exemple, sur la figure 4.17, la branche de l'arbre PN_2 mise à jour par la triangulation récupère le brin indiqué à l'étape précédente, b, et l'orbite indiquée par le journal d'historique de PI_{2b} pour la triangulation, $\langle 0, 2\rangle$ (figure 4.7a).

Note : en cas de suppression d'une opération, on peut ne plus pouvoir appliquer certaines opérations. C'est le cas de $6 - Triangulation$ sur la figure 4.8, car le journal d'historique

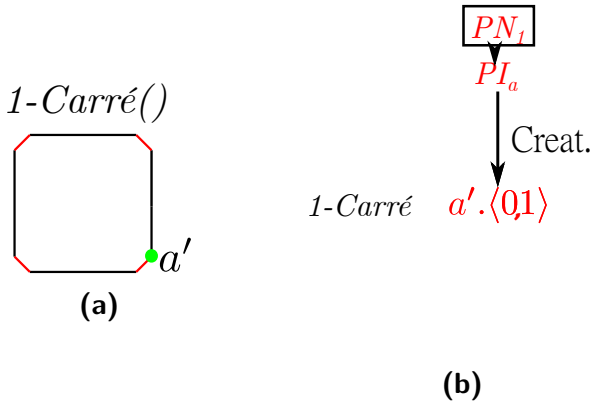


Figure 4.16 : Arbre d'appariement de PN_1 après l'étape 1. a : modèle. b : arbre.

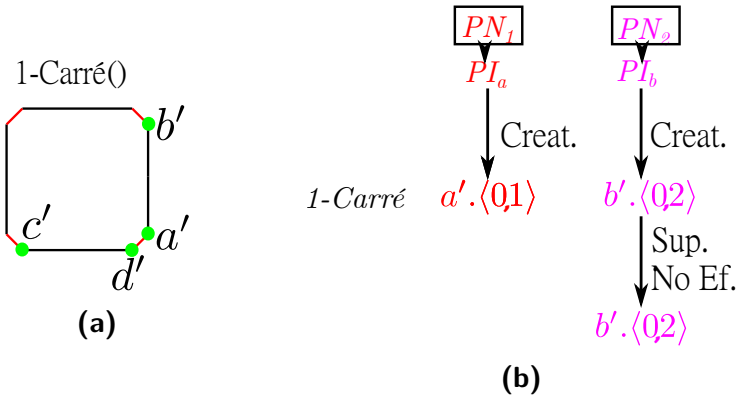


Figure 4.17 : Arbre d'appariement de PN_1 après l'étape 1. a : modèle. b : arbre.

de PI_{5b} montre qu'une scission a été supprimée quand on a supprimé $2-Triangulation$ (Annexe B.1). Cependant, sur cette figure, on constate que $7-Coloration$ n'a pas non plus été appliquée. C'est compréhensible pour la coloration attachée à PN_6, le journal de PI_{6g} comprenant un "Creat." lié à $2-Triangulation$ dans son journal d'historique, mais pas pour PN_7, car PI_{7a} y associait "No Eff.". Cependant, $6-Triangulation$ n'ayant pas pu être appliquée, on la considère comme supprimée. On recalculera alors les journaux d'historique en conséquence et celui de PI_{7a} deviendra le journal présenté sur la figure 4.18. Son arbre d'appariement faisant maintenant référence à "Sup. Split" pour $6-Triangulation$, l'opération ne sera pas appliquée.

4.2.3 Opérations déplacées

En cas de changement d'ordre d'une opération, on donnera le label "Chang." à la branche, comme indiqué plus haut. Les feuilles seront elles constituées de l'ensemble des orbites désignées par :

— le type d'orbite indiqué par le journal de bord recalculé ;

— l'un des brins composant l'orbite avant le changement.

La figure 4.19 présente notre modèle après l'étape 1 (qui a consisté à appliquer $1-Carré$), ainsi que les arbres correspondant aux différents identifiants persistants composant PN_5 pour cette étape).

L'étape 2 consiste à déplacer l'opération $2-Triangulation$. C'est à ce moment-là qu'on recalcule les journaux d'historique des noms impactés (section 4.1.4). On obtient donc trois feuilles après l'étape 1, et six après l'étape 2 :

— l'orbite $a'.\langle 0\rangle$, composée des brin a' et b', et qui pointera donc vers $a'.\langle 0\rangle$ et $b'.\langle 0\rangle$ après le changement ;

— deux orbites $b'.\langle 0\rangle$, composées des brin a' et b', et qui pointeront donc, l'une vers chacune vers $a'.\langle\rangle$ et $b'.\langle\rangle$, et l'autre vers $a'.\langle 0\rangle$ et $b'.\langle 0\rangle$ après le changement.

Les six nouvelles branches associées à ces feuilles porteront le label "Dépl." (figure 4.20).

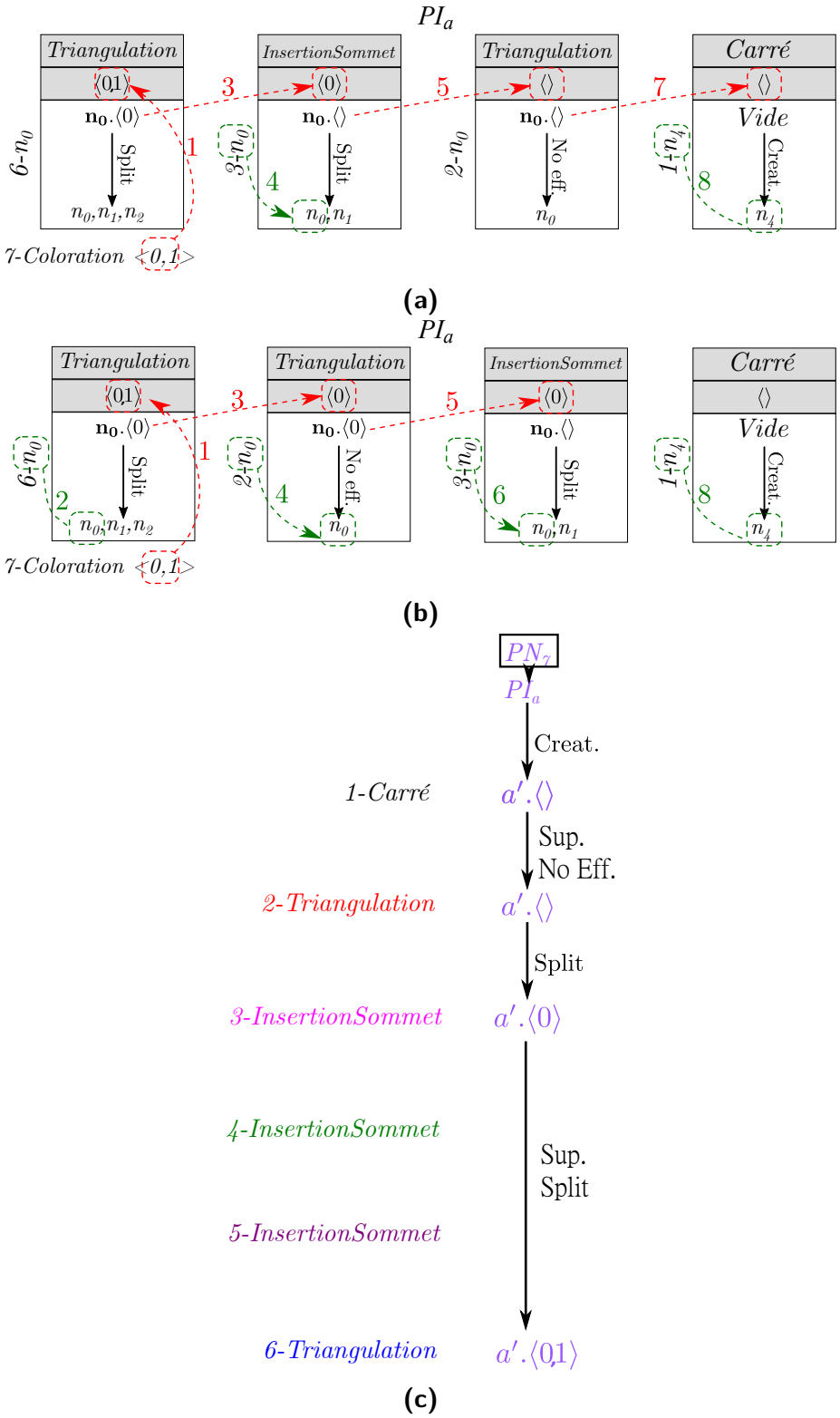


Figure 4.18 : a : Journal d'historique hypothétique de PI_{7a}. b : Journal d'historique réél de PI_{7a}. 96 Arbre d'appariement hypothétique de PN_7

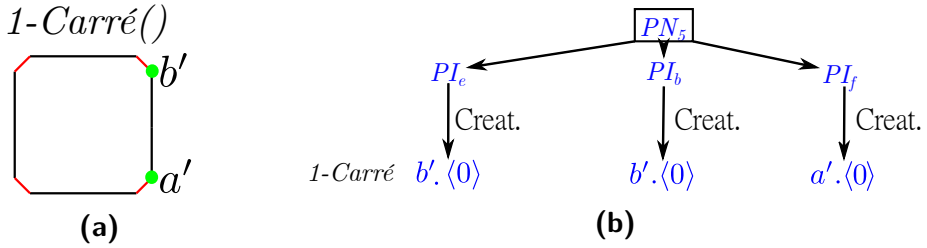


Figure 4.19 : Arbre d'appariement de PN_5 après l'étape 1. a : modèle. b : arbre.

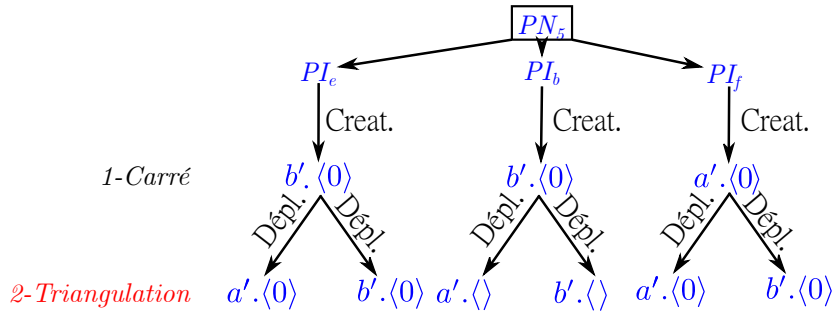


Figure 4.20 : Arbre d'appariement PN_5 après l'étape 2

4.2.4 Opérations ajoutées

Penchons-nous maintenant sur le cas des opérations ajoutées. Comme indiqué précédemment, on ne peut pas modifier les différents journaux d'historique des noms persistants pour ces opérations, car on ne peut pas savoir à l'avance quelles orbites seront affectées.

Pour en tenir compte, nous allons donc introduire la notion d'*arbre virtuel* :

Definition 12 (*Arbre virtuel*)

Un arbre virtuel est composé :

- *d'une racine, le type d'orbite auquel il est attaché*
- *de branches portant un type de modification*
- *de feuilles, composées chacune d'un brin filtré par la règle appelée par l'opération ajoutée et permettant d'identifier l'orbite affectée, une fois associé au type d'orbite de la racine, ou de la mention Vide si on supprime une orbite.*

Quand une opération est ajoutée, on procède en trois étapes.

4.2.4.1 Étape 1 : Identification des orbites potentiellement impactées

Pour tous les arbres d'appariement encore utilisés, on consulte les journaux d'historiques associés à leurs identifiants persistants, et on liste les types d'orbites qui n'ont pas été utilisés pour la mise à jour de l'arbre (c'est-à-dire ceux correspondant aux opérations qui n'ont pas encore été effectuées), et ceux portés par les feuilles de chaque arbre.

4.2.4.2 Étape 2 : Création des arbres virtuels

Dans un deuxième temps, on s'appuie sur ces types d'orbites pour créer les arbres virtuels. On va créer un arbre par type d'orbite différent listé, soit 8 arbres par opérations en dimension 2, et 16 en dimension 3.

Chaque arbre sera composé de :

— sa racine : le type d'orbite $\langle o \rangle$ qu'il concerne ;

— ses branche, portant le label du type de modification affectant l'arc du journal de bord de la règle appelée par l'opération pour le type $\langle o \rangle$. S'il y a plusieurs graphes dans cet extrait de journal de bord, on choisira celui ou ceux qui auront un nœud "hook" [1] dans leurs nœuds d'origine **et** dans leurs nœuds extrémité ;

— ses feuilles, composées chacune d'un brin ou de la mention *Vide* dans le cas d'une suppression. Ce brin est choisi parmi ceux qui sont filtrés par les nœuds extrémité du graphe extrait du journal de bord ayant permis d'étiqueter la branche. Le brin permet de savoir quelle entité est modifiée.

4.2.4.3 Étape 3 : Utilisation des arbres virtuels

Le principe est de répéter la même procédure à chaque mise à jour des arbres d'appariement restants, *en incluant l'étape à laquelle on a ajouté l'opération* (pour pouvoir mettre à jour les arbres d'appariement le cas échéant, s'ils sont directement impactés).

Afin de réaliser la mise à jour d'un arbre d'appariement par l'opération ajoutée, on procède en plusieurs étapes :

1. Le ou les nœuds "hook" sont ceux qui correspondent aux brins sélectionnés par l'utilisateur pour appliquer la règle (ils sont donc situés dans la partie gauche de cette règle). Ces nœuds "hook" sont spécifiés lors de la création de la règle. Dans les différentes règles que nous avons présentées jusqu'ici, tous les nœuds de gauche sont des hooks. Les nœuds hooks sont équivalents entre eux, en choisir un au lieu d'un autre n'a donc pas d'influence sur le résultat de l'application de la règle.

1. On détermine si l'entité feuille de l'arbre d'appariement est égale à l'une des entités référencées par un des brins feuilles de l'arbre virtuel associé au type d'orbite formant sa racine (c'est-à-dire, si l'entité feuille de l'arbre d'appariement correspond à une entité modifiée par l'opération ajoutée).

   Sinon, il n'y a pas à faire de mise à jour, et on ne suit pas les étapes suivantes.

2. On vérifie si l'opération ajoutée a déjà induit une mise à jour de l'arbre d'appariement pour le type d'orbite indiqué par sa racine.

   Si tel est le cas, on ne met pas à jour l'arbre d'appariement, car cela reviendrait à prendre en compte deux fois l'opération pour le même type d'orbite. On ne suit pas l'étape suivante.

3. Enfin, si les deux pré-conditions sont réunies, on réalise la mise à jour de l'arbre d'appariement.

   Pour cela, on ajoutera un nœud à l'arbre d'appariement à l'étape où se jouait l'opération ajoutée (si la mise à jour se fait immédiatement à l'ajout de l'opération, cela revient donc à ajouter une feuille). La branche menant à cette feuille sera étiquetée "*Aj.+Type Modif.*" avec *Type Modif.* le type de modification indiqué par l'arbre virtuel, et le nœud sera composée du brin support du nœud le précédant, et du type d'orbite indiqué par l'arbre virtuel. Dans le cas de la scission, on créera également au niveau de cet ajout une série de feuilles parallèles, qui utiliseront chacune comme brin support l'un des brins de l'arbre virtuel.

Un arbre d'appariement peut donc être mis à jour plusieurs fois par l'ajout d'une même opération, dans la limite d'une fois par type d'orbite.

4.2.4.4 Application à un exemple

Considérons l'exemple présenté sur la figure 4.21a. On crée un pentagone, sur lequel on insère une arête ($PN_1 = PI_{a1}.\langle 0, 2\rangle$, $PN_2 = PI_{b1}.\langle 0, 2\rangle$), avant de colorer l'une des faces obtenues ($PN_3 = PI_{a2}.\langle 0, 1\rangle$).

Au rejeu, on ajoute une insertion d'arête (portant le numéro "A1" : "A" pour l'Ajout d'une opération et "1" pour indiquer qu'il s'agit de la première opération ajoutée). Cela entraîne la scission de la face colorée, et donc le rejeu de la coloration pour chacune des faces issues de la scission (figure 4.21b).

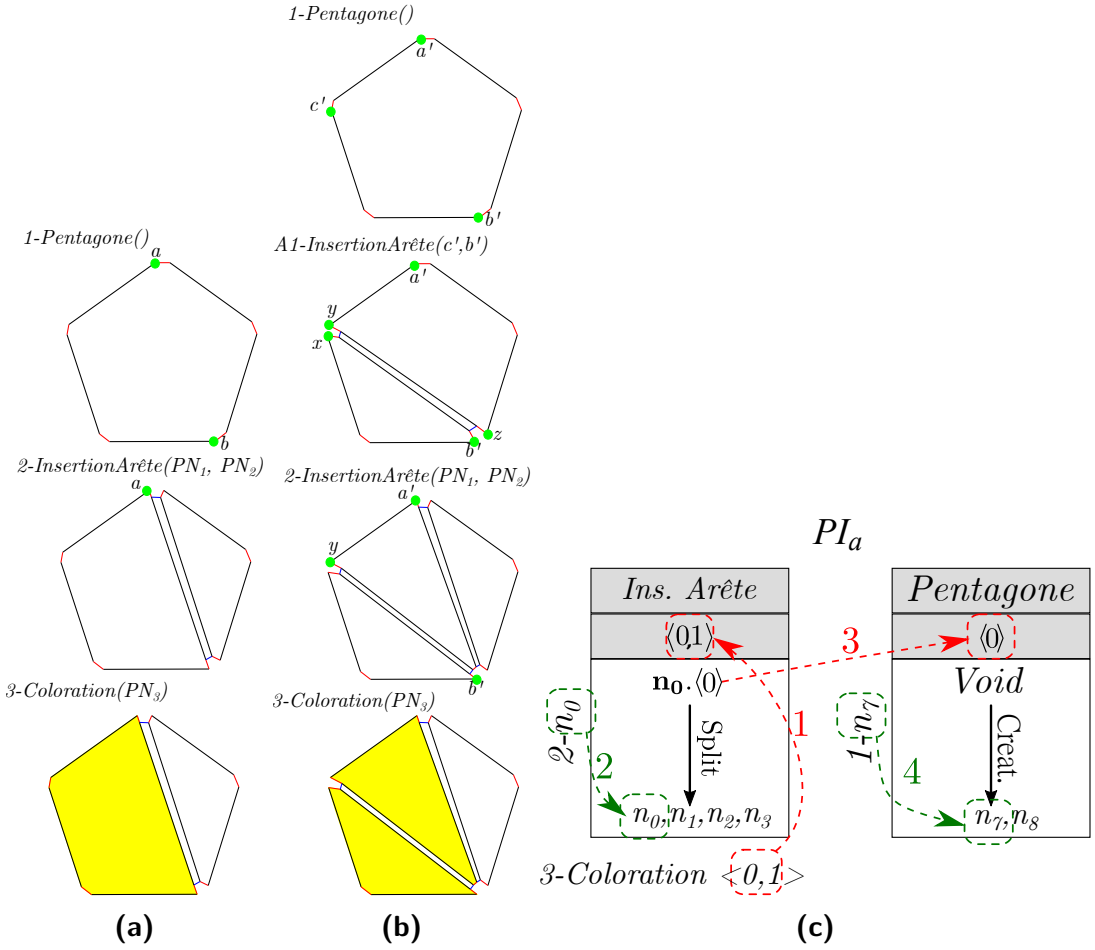


Figure 4.21 : Fil Rouge de l'ajout d'opérations - Jeu Initial et Rejeu, Journal d'historique associé à PN_3

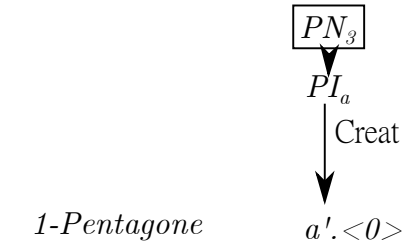


Figure 4.22 : Fil Rouge de l'ajout d'opérations - Première mise à jour de l'arbre d'appariement

La figure 4.21c présente le journal d'historique associé à PN_3 *via* PI_{a2}.

Détaillons maintenant le rejeu. Dans un premier temps, on rejoue à l'identique la création du Pentagone. On met à jour l'arbre d'appariement de PN_3 (figure 4.22).

Dans un deuxième temps, on ajoute l'insertion d'arête $A1$. On s'intéresse à son impact sur l'arbre d'appariement lié à PN_3. Cet arbre d'appariement nous indique qu'il faut prendre en compte le type d'orbite $\langle 0 \rangle$, qui compose sa feuille, et son journal d'historique nous indique qu'il faut prendre en compte le type d'orbite $\langle 01 \rangle$, attaché à l'opération 2-InsertionArête, qui n'est pas encore appliquée.

La figure 4.23 représente la règle d'insertion d'arête (fig. 4.23a), des extraits de son journal de bord pour les types d'orbites $\langle 01 \rangle$ (fig. 4.23b) et $\langle 0 \rangle$ (fig. 4.23c), et les deux arbres virtuels (fig. 4.23d-4.23e) que l'on construit grâce à eux, en sélectionnant des brins filtrés par n_0 ou n_1 (les nœuds "hook") ; les brins b', x, y et z sont représentés sur la figure 4.21b. Dans l'arbre représenté sur la figure 4.23e, deux des branches (pointant vers z et b') proviennent de l'extrait de journal de bord de $\langle 0 \rangle$ pointant de n_0 vers n_0, et les deux autres, de l'extrait pointant de n_1 vers n_1.

À cette étape, l'arbre d'appariement de PN_3 a pour feuille $a'.\langle 0 \rangle$ (fig. 4.22). Or, $a'.\langle 0 \rangle = y.\langle 0 \rangle$. On met donc à jour l'arbre d'appariement, avec une nouvelle branche portant le label "Aj. No Eff.", le type de modification porté par l'arbre virtuel, et une feuille conservant le type d'orbite $\langle 0 \rangle$, porté par le brin a' (y n'est là que pour identifier l'orbite affectée par la règle), ce qui est présenté sur la figure 4.24. N'ayant plus à utiliser l'arbre virtuel lié à $\langle 0 \rangle$, on le supprime.

Ensuite, on rejoue l'insertion d'arête 2. Grâce au journal d'historique de PN_3, on met à jour l'arbre d'appariement (figure 4.25a). On obtient ainsi l'orbite $a'.\langle 01 \rangle$. Or, on n'a

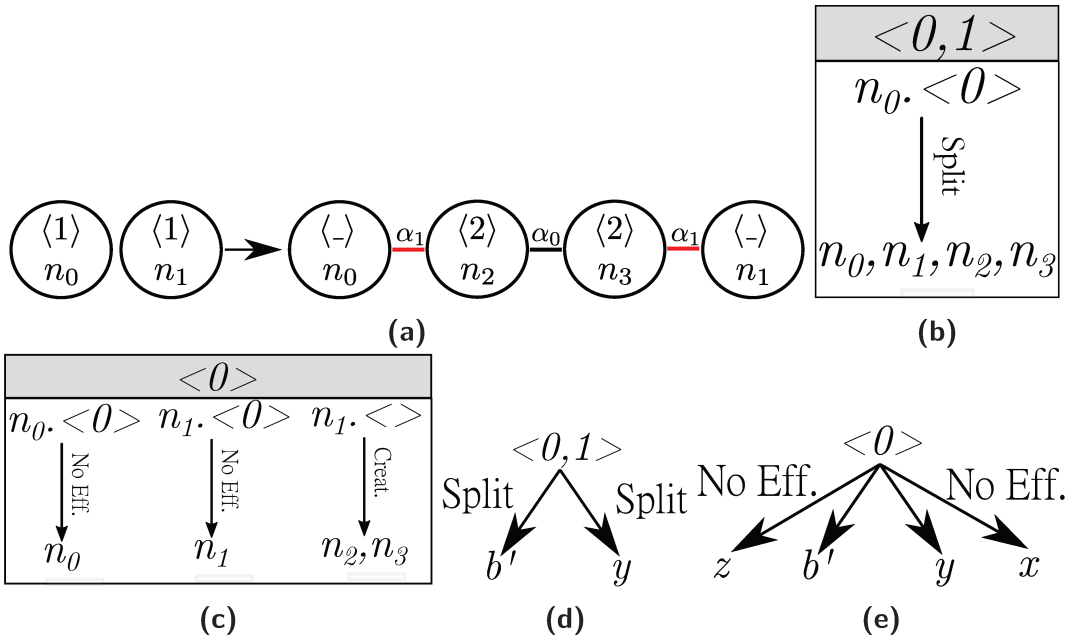


Figure 4.23 : Règle d'insertion d'arête : règle, extraits de journal de bord et arbres virtuels

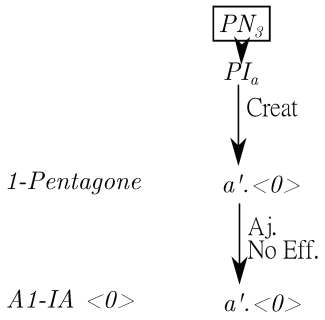


Figure 4.24 : Fil Rouge de l'ajout d'opérations - Deuxième mise à jour de l'arbre d'appariement

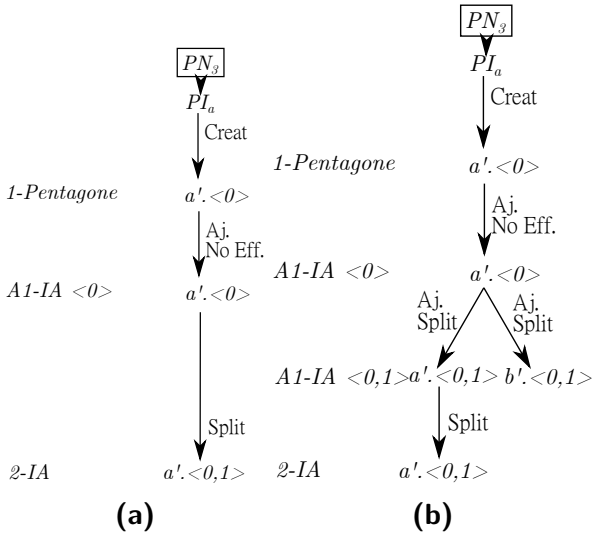


Figure 4.25 : Fil Rouge de l'ajout d'opérations - Dernières mises à jour de l'appariement

pas encore réalisé la mise à jour de l'arbre pour l'opération ajoutée $A1$ et le type d'orbite $\langle 01 \rangle$. Il faut déterminer si cette mise à jour est nécessaire. $a'.\langle 01 \rangle = y.\langle 01 \rangle$, il faut réaliser la mise à jour. Pour cela, on ajoute rétrospectivement un nœud $a'.\langle 01 \rangle$ pour l'opération ajoutée dans l'arbre (figure 4.25b). Cependant, l'arbre virtuel de la figure 4.23d indique qu'il y a scission pour ce type d'orbite. En parallèle de ce nœud, on ajoute donc également une feuille $b'.\langle 01 \rangle$. Le nœud et la feuille sont tous les deux pointés par une branche "Aj. Split", la modification étant une scission, ce qui donne l'arbre d'appariement présenté sur la figure 4.25b. N'ayant plus à utiliser l'arbre virtuel, on le supprime.

La coloration sera donc appliquée sur deux orbites, $a'.\langle 01 \rangle$ et $b'.\langle 01 \rangle$, d'où le résultat présenté sur la figure 4.21b.

4.2.5 Application à notre fil rouge

4.2.5.1 Étape 1 - Rejeu de 1-Carré

La première étape consiste à rejouer $1 - Carré$. On lit alors simplement les journaux de bord (Annexe B). Étant donné que, lors de la création du carré, n_3 filtre a', n_4 b', n_5 d' et n_6 c' (figure 4.26a), on obtient les arbres d'appariement de la figure 4.26b.

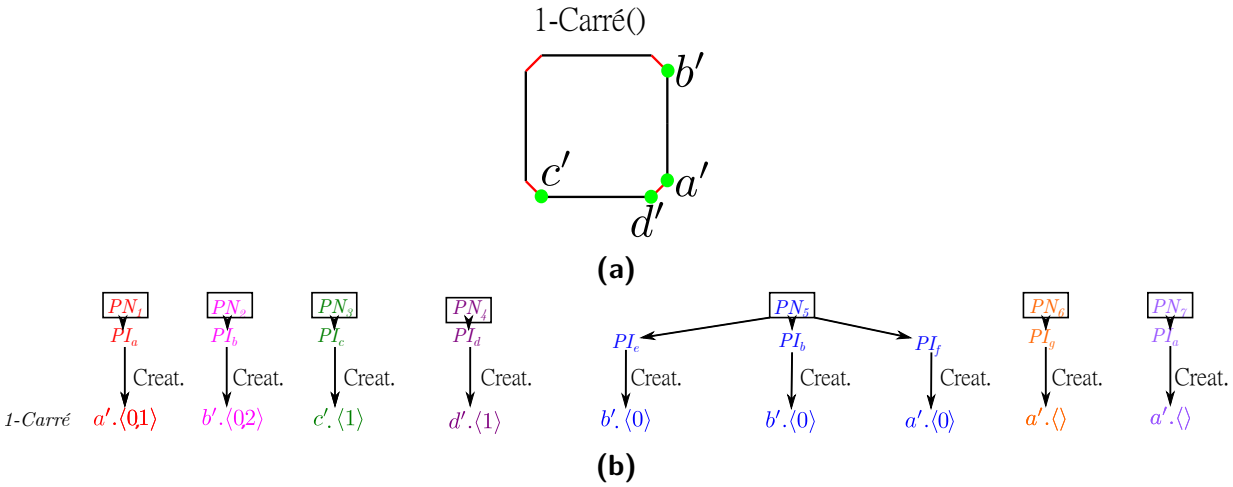


Figure 4.26 : Étape 1. a. Résultat. b. Arbres.

4.2.5.2 Étape 2 - Déplacement de 2-Triangulation

La deuxième étape consiste à déplacer $2-Triangulation$. Comme indiqué précédemment, c'est à cette étape que l'on recalcule les journaux d'historique impactés par cette modification, à savoir ceux des identifiants de PN_4, PN_5, PN_6 et PN_7. Les nouvelles branches de ces arbres d'appariement portent le label "Dépl.", et sont composées de tous les brins qui composaient leurs orbites à l'étape précédente.

On obtient alors :

— pour PN_4, l'orbite $d'.\langle 1\rangle$, composée des brins a' et d', ce qui donne les deux feuilles $a'.\langle 1\rangle$ et $d'.\langle 1\rangle$;

— pour PN_5, deux orbites $b'.\langle 0\rangle$, composées des brins a' et b', et une orbite $a'.\langle 0\rangle$, composée de ces mêmes brins, ce qui donne une feuille $a'.\langle 0\rangle$ et une feuille $b'.\langle 0\rangle$ pour deux des arbres, et une feuille $a'.\langle\rangle$ et une feuille $b'.\langle\rangle$ pour le troisième ;

— pour PN_6 et PN_7, l'orbite $a'.\langle\rangle$, n'est composée que du brin a', ce qui donne la feuille $a'.\langle\rangle$.

Les arbres liés à PN_2 et PN_3, eux, sont mis à jour en prenant en compte que $2-Triangulation$ est supprimée de leur point de vue. On garde donc le brin de leur dernière feuille, b' pour PN_2 et c' pour PN_3, auquel on associe le type d'orbite qu'ils auraient dû avoir après la triangulation (indiqué par le journal d'historique), $\langle 0,2\rangle$. Enfin, on fait porter aux deux branches le label "Sup." en plus du type de modification indiqué par chacun des deux journaux d'historique pour l'opération $2-Triangulation$. Ces résultats

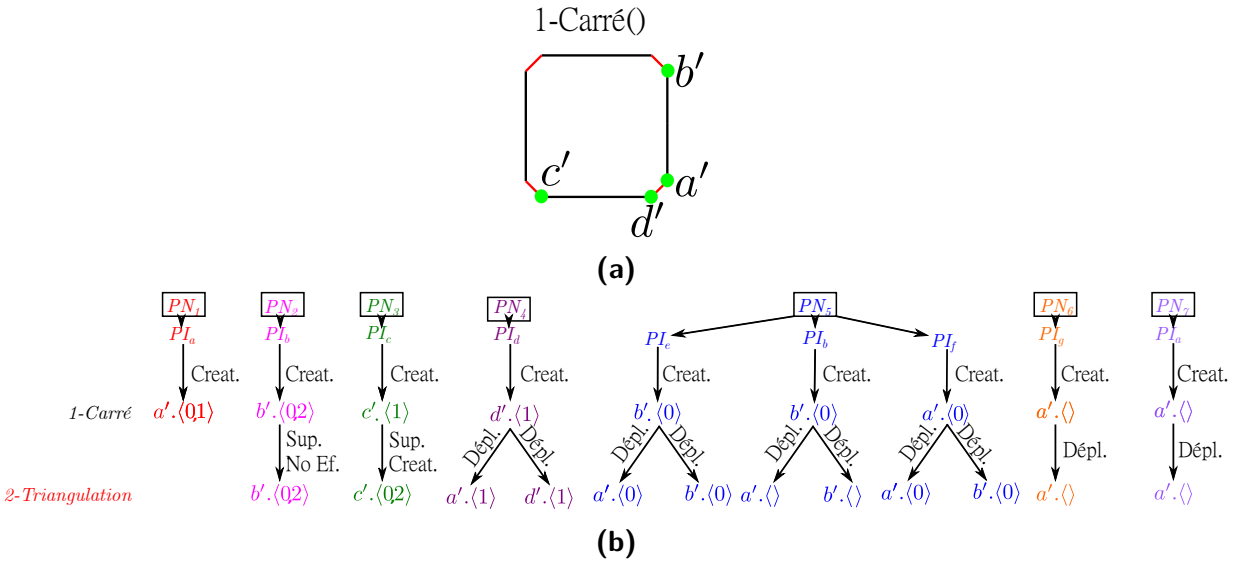


Figure 4.27 : Étape 2. a. Résultat. b. Arbres.

sont présentés sur la figure 4.27.

4.2.5.3 Étape 3 - Rejeu de 3-InsertionSommet

À cette étape, on rejoue $3 - InsertionSommet$. Celle-ci doit être rejouée sur $b'.\langle 0, 2\rangle$. Aucune des branches n'est interdite par le paramétrage (qui, pour rappel, n'interdit que les branches "Sup. Creat." et "Sup. Split"), on peut donc la rejouer (résultat sur la figure 4.28).

Pour PN_1, la règle est ajoutée. Pour l'orbite $\langle 0, 1\rangle$, le journal de bord de l'insertion de sommet indique $n_0.\langle 0, 1\rangle \rightarrow n_0, n_1$, la branche portant le label "Modif.", ce qui donne l'arbre virtuel présenté sur la figure 4.28b. Or, $b'.\langle 0, 1\rangle = a'.\langle 0, 1\rangle$, on fait donc pointer la nouvelle branche de l'arbre vers $a'.\langle 0, 1\rangle$, avec le label "Aj. Modif." (figure 4.28).

Pour les autres noms persistants, l'opération est traitée comme une opération rejouée. Deux arbres seront impactés, celui lié à PN_7 et l'un de ceux liés à PN_5, qui sont les seuls à comprendre la valeur 3 (le numéro de l'opération) dans les journaux d'historique de leurs identifiants, PI_{7a} et PI_{5b}. En se basant sur ces derniers pour l'extrait de journal d'historique lié à 3, on peut mettre à jour les arbres.

Pour PI_{5b}, le journal indique $n_0.\langle\rangle \rightarrow n_0, n_1$, label "Split". On prendra donc comme nouveau brin représentant de l'orbite le brin filtré par n_0 à droite, et étant issu de la même

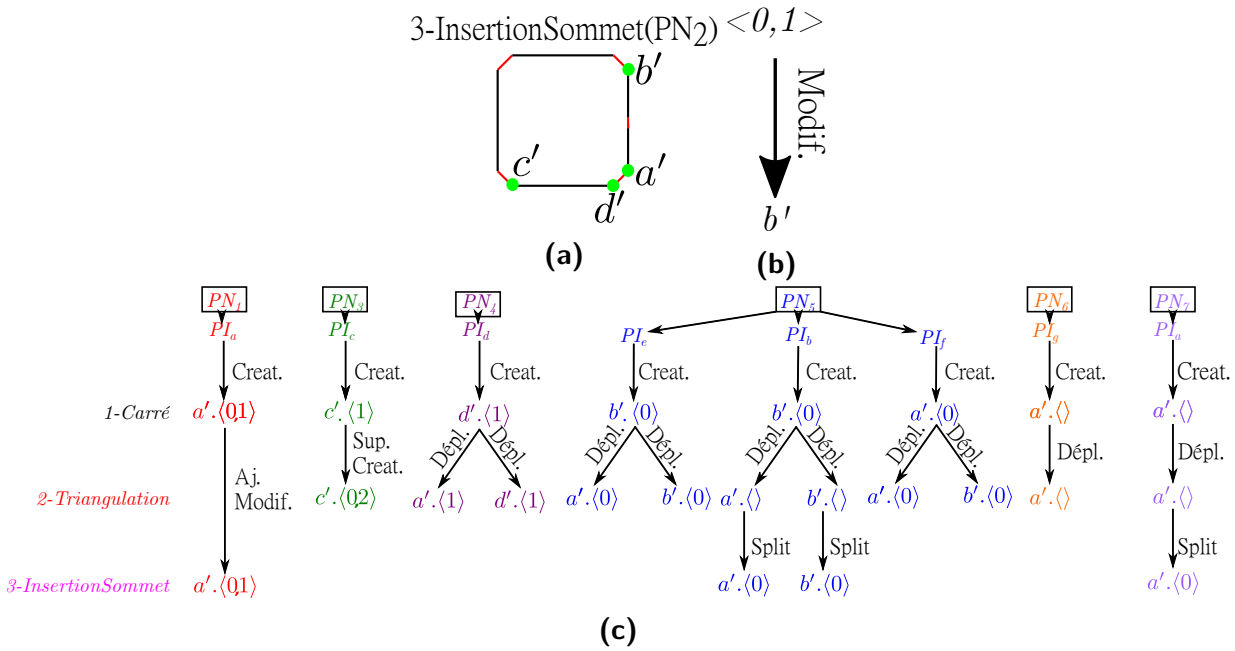


Figure 4.28 : Étape 3. a. Résultat. b. Arbres.

copie que le brin filtré par n_0 à gauche (c'est-à-dire, le même brin, comme on est en présence du même nœud). Il y a deux branches à étudier, on aura donc, comme brins représentants, a' pour la première feuille et b' pour la seconde, associée à l'orbite $\langle 0 \rangle$, l'orbite indiquée par le journal d'historique pour l'opération 3 (figure 4.28).

Pour PI_{7a}, le journal nous indique également $n_0.\langle\rangle \rightarrow n_0, n_1$, label "Split". On aura donc comme feuille $a'.\langle 0 \rangle$, $\langle 0 \rangle$ étant à nouveau l'orbite indiquée par le journal d'historique (figure 4.28).

4.2.5.4 Étape 4 - Rejeu de 4-InsertionSommet

À cette étape, on rejoue $4 - InsertionSommet$ sur $c'.\langle 0, 2 \rangle$. Cependant, son arbre comprend un label "Sup. Creat.". On ne peut donc pas l'appliquer et elle est considérée comme supprimée.

L'opération n'apparaissant dans aucun des journaux d'historique des noms restants à apparier, cela n'a pas d'impact sur les autres opérations de notre modèle (figure 4.29).

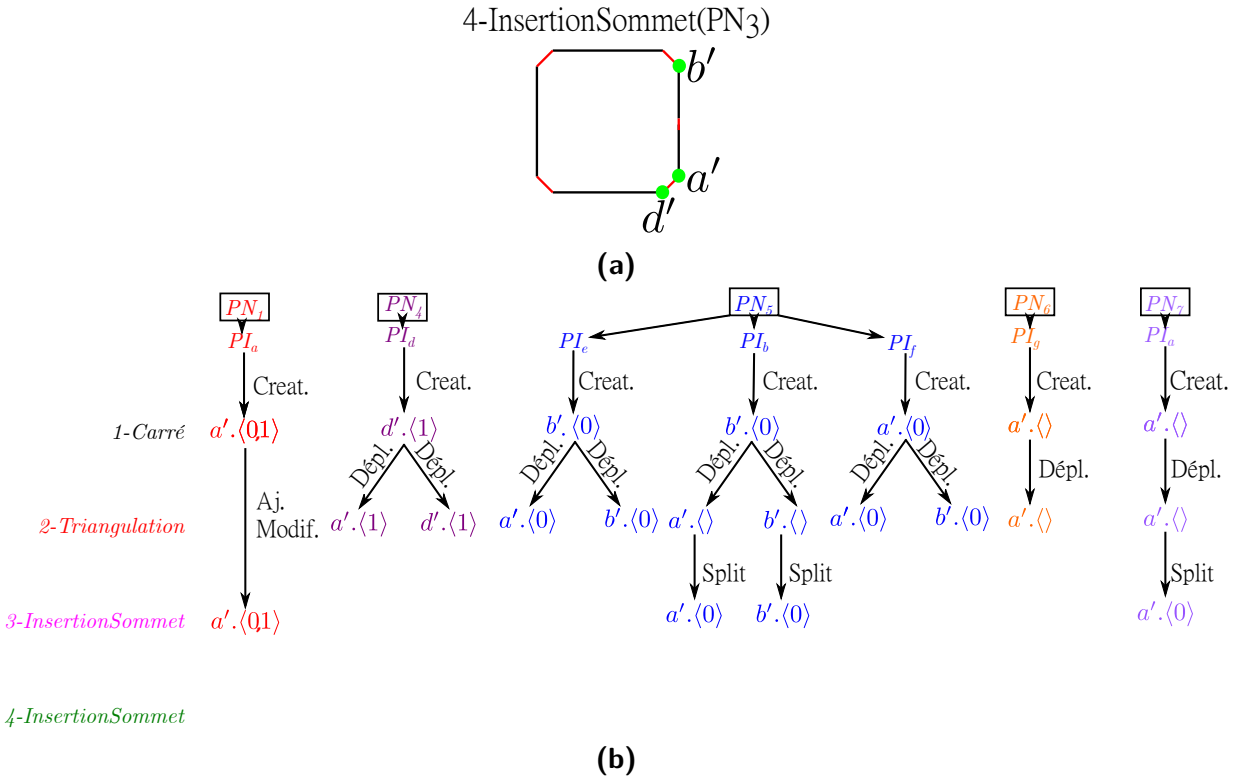


Figure 4.29 : Étape 4. a. Résultat. b. Arbres.

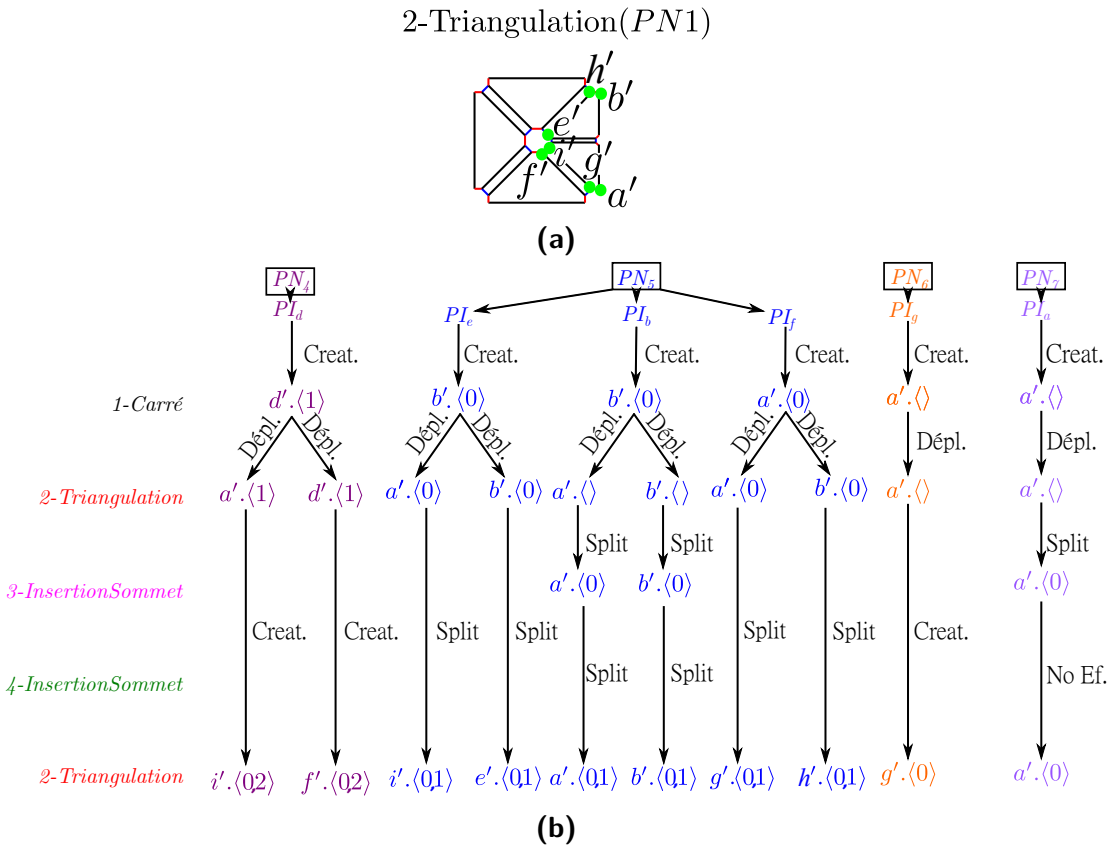


Figure 4.30 : Étape 5. a. Résultat. b. Arbres.

4.2.5.5 Étape 5 - Rejeu de 2-Triangulation

À cette étape, on rejoue $2-Triangulation$ sur $a'.\langle 0,1\rangle$. Toutes les branches sont impactées, et on les met à jour de la même manière qu'on avait effectué la mise à jour à l'étape 3.

À titre d'exemple et pour présenter un cas où les brins support de l'orbite à l'origine de la branche et à son arrivée ne sont pas les mêmes, nous allons détailler la mise à jour de l'arbre de PN_4. Pour l'opération 2, le journal d'historique nous indique : $n_0.\langle 1\rangle \rightarrow n_1, n_2$ label "Creat.", avec l'en-tête $\langle 0,2\rangle$. Avant l'étape, l'arbre de PN_4 a deux feuilles, $a'.\langle 1\rangle$ et $d'.\langle 1\rangle$ (figure 4.29). Le brin créé par n_2 par copie de a' *via* le nœud n_1 est i', et celui créé par copie de d' est f'. On a donc comme feuilles les orbites $i'.\langle 0,2\rangle$ et $e'.\langle 0,2\rangle$. Le résultat est présenté sur la figure 4.30.

4.2.5.6 Étape 6 - Rejeu de 5-InsertionSommet

On rejoue $5 - InsertionSommet$ sur $i'.\langle 0,2\rangle$ et sur $f'.\langle 0,2\rangle$. i' et f' font partie de la même arête et apparient le même nom persistant, PN_4 (figure 4.30), on applique donc l'insertion de sommet une seule fois. On réalise les mises à jour comme on l'a fait aux étapes précédentes, ce qui donne le résultat présenté à la figure 4.31, seul l'un des identifiants de PN_5, $PI_5 f$ et celui de PN_6, PI_{6g} étant affectés car ayant trace de l'opération dans leurs journaux d'historique. La feuille de l'arbre PI_f contenant h' ne peut pas être mise à jour, h' n'étant en réalité pas affecté par l'insertion. On la met donc à jour comme si l'opération était supprimée.

4.2.5.7 Étape 7 - Rejeu de 6-Triangulation

On rejoue $6 - Triangulation$ sur $i'.\langle 0,1\rangle$, $e'.\langle 0,1\rangle$, $a'.\langle 0,1\rangle$, $b'.\langle 0,1\rangle$, $j'.\langle 0,1\rangle$ et sur $h'.\langle 0,1\rangle$. Cela définit deux faces, celle contenant i', a' et j', et celle contenant b', h' et e'. On applique donc l'opération sur ces deux faces, et on met à jour les deux arbres restants (figure 4.32).

4.2.5.8 Étape 8 - Rejeu de 7-Coloration

À cette étape, on rejoue $7 - Coloration$, sur $g'.\langle 0,1\rangle$ pour PN_6 et $a'.\langle 0,1\rangle$ pour PN_7. L'opération utilise les deux derniers arbres restants, il n'y en a donc plus à mettre à jour. Le résultat est présenté sur la figure 4.33.

Au cours de ce chapitre, nous avons vu comment utiliser nos différents outils afin de réaliser un rejeu avec édition. Il reste maintenant à positionner notre méthode par rapport à celles déjà existantes.

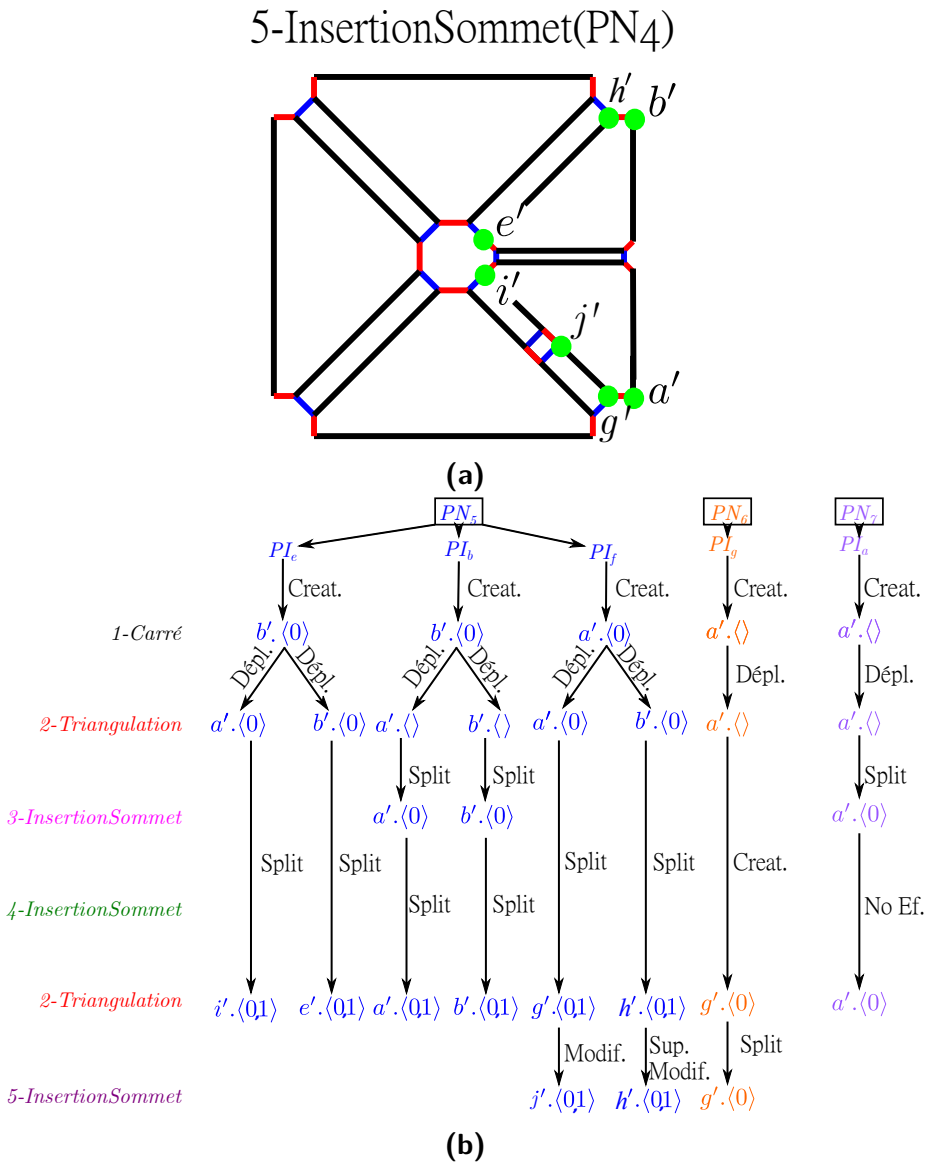


Figure 4.31 : Étape 6. a. Résultat. b. Arbres.

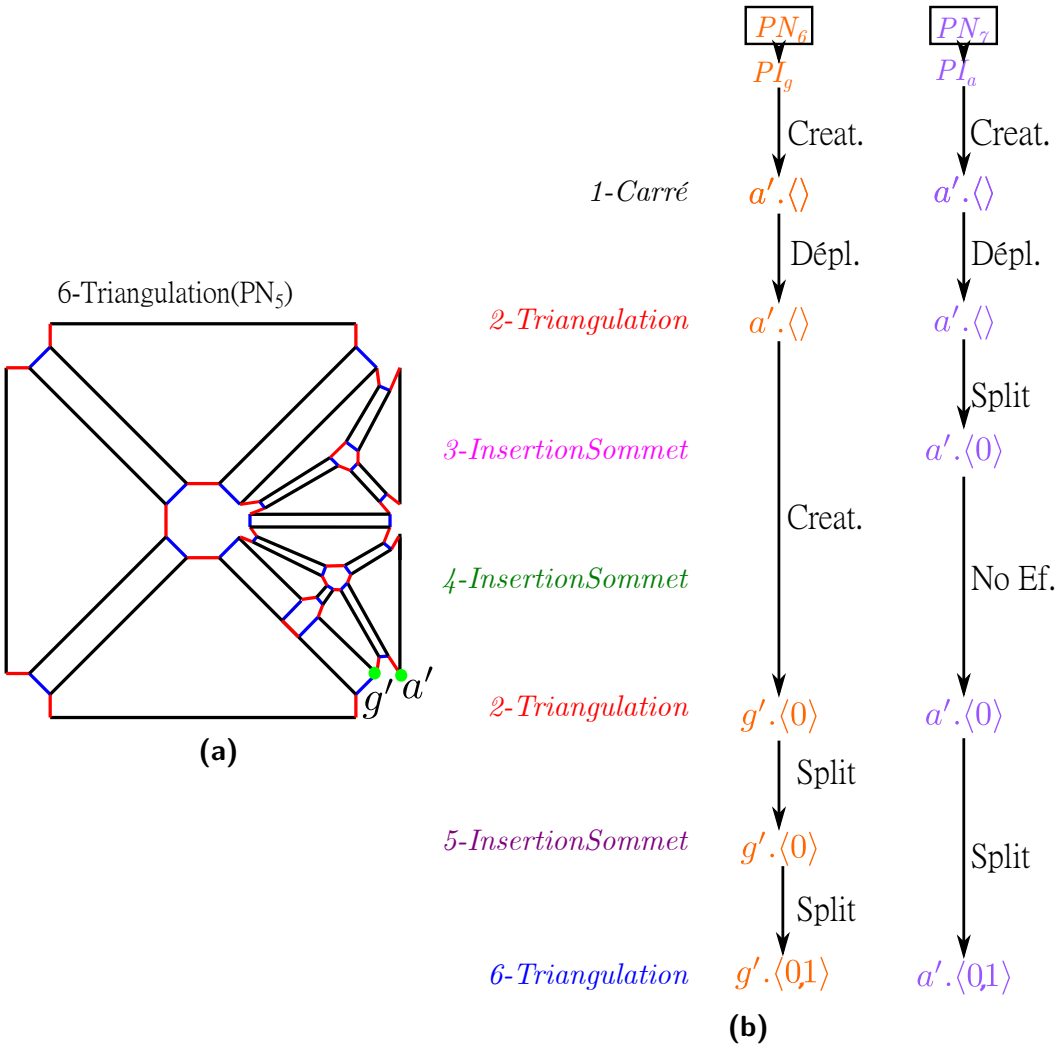


Figure 4.32 : Étape 7. a. Résultat. b. Arbres.

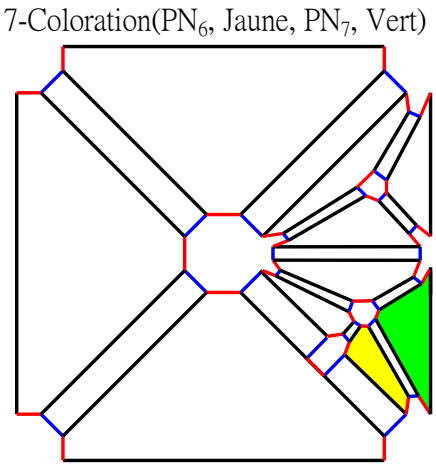


Figure 4.33 : Étape 8, Résultat

I consider experience, experience

Loki, Avengers Infinity War

5

Résultats et comparaison aux autres méthodes

Sommaire

Dans les deux chapitres précédents, nous avons détaillé notre méthode permettant d'éditer une spécification, non seulement en modifiant ses paramètres géométriques, mais aussi en supprimant, en ajoutant ou en modifiant l'ordre de ses opérations. À notre connaissance, les travaux antérieurs ne se sont intéressés qu'au problème du changement de paramètres, ou ne donnent que peu d'informations sur la manière dont sont ajoutées, supprimées ou déplacées les opérations. C'est donc le premier point sur lequel notre méthode se différencie.

Nous avons proposé, dans une approche novatrice, de combiner spécifications paramétriques et règles de transformation de graphes en se basant sur la bibliothèque Jerboa et sur l'utilisation des G-Cartes. Dans ce contexte, nous avons étudié et proposé un système de nommage tirant profit des spécificités des règles.

Nous allons comparer dans ce chapitre notre méthode aux méthodes existantes, notamment par rapport aux concepts communs (section 2.4.2.1), et réaliser une estimation et une comparaison théorique de la taille des structures de données nécessaires au nommage persistant.

5.1 Retour sur les concepts communs

Dans l'étude [MP02], les auteurs présentent un certain nombre de "concepts communs" que l'on retrouve dans la plupart des approches traitant du problème de nommage persistant.

5.1.1 Entités invariantes et contingentes, et nommage de ces entités

Dans la section 2.4.2.1, nous avons expliqué que la plupart des méthodes classent les entités en deux groupes : les entités invariantes, caractérisées directement grâce aux opérations de modélisation, et les entités contingentes, issues de l'interaction entre des géométries existantes.

La plupart des méthodes utilisent ensuite deux types de nommage différents, un pour les entités invariantes, et un pour les entités contingentes, ces deux types pouvant eux-mêmes être encore divisés en plusieurs types, généralement un par dimension d'entité. De plus, comme nous l'avions vu à la section 2.4.2.2, ces systèmes de nommage sont généralement basés sur l'entité de dimension $n-1$ pour un modèle en dimension n (les faces en dimension 3), ce qui les rend dépendants de la dimension du modèle.

Avec notre méthode, nous avons voulu mettre en place un nommage homogène (au sein d'un modèle, un même nommage pour toutes les entités, indépendamment de leur dimension) et général (le nommage ne dépend pas de la dimension du modèle). De plus, les entités étant toutes directement impactées par les règles qui les filtrent, les concepts d'entités invariantes et contingentes n'ont pas d'influence sur la manière dont est réalisé notre nommage pour le moment. Cependant, cela pourra changer dans le futur avec l'étoffement du système de nommage pour intégrer l'utilisation de règles non-élémentaires (voir section 6.2.1).

5.1.2 Historique dans la nomination

Le concept commun suivant est l'historique des noms. Celui-ci permet, d'une manière ou d'une autre, d'associer au nom d'une entité les différentes transformations ayant affecté cette entité.

Dans notre méthode, l'historique est réalisé grâce à trois éléments :

— Le nom persistant, qui tient compte de toutes les opérations ayant affecté une entité *via* les identifiants persistants de ses différents brins. Capoyleas et *al.* [CCH96] propose une approche qui incorpore également une forme d'historique dans le nom : le nom d'une entité extrudée est par exemple basé sur celui de l'entité qui en est à l'origine.

— Les arbres d'appariement, notion en partie comparable aux travaux de Kripac [Kri95], qui suivent l'évolution de chaque face afin, au rejeu, de pouvoir trouver un apparie-

ment en comparant les arbres du rejeu et du jeu initial. Dans nos travaux, nous suivons uniquement les entités utilisées par la spécification paramétrique, et uniquement au rejeu, en référençant directement les entités appariées à un nom persistant.
— Les journaux d'historique, qui permettent de retenir toutes les opérations ayant affecté une entité, et la manière dont elles l'ont affectée.

5.1.3 Architecture à trois couches

Enfin, le dernier concept commun concerne l'architecture à trois couches : géométrie, nommage et spécification paramétrique.

Dans notre méthode, nous retrouvons bien cette structure, avec
— la couche géométrique, composée du modèle réalisé en utilisant une G-Carte,
— la couche de la spécification paramétrique, qui contient la liste des opérations et leurs paramètres,
— la couche de nommage, composée des noms persistants, des journaux d'historique et des arbres d'appariement.

5.1.4 Appariement, local ou global ?

Un autre élément, dont nous n'avons pas parlé dans la section 2.4.2.1 est la notion d'appariement global ou local. En fonction de la littérature dans lesquels elle apparaît, cette notion est parfois considérée comme un concept commun, parfois non, toute méthode ayant un appariement d'un seul type, local ou global.

L'appariement local consiste à tenter d'apparier l'entité initiale avec un ensemble d'entités rejouées. On parcourt alors cet ensemble pour trouver une entité correspondante à l'initiale. On trouve parmi les tenants de ces méthodes Capoyleas et Chen [CCH96] et Wu [WZZ01]. Dans cette approche, on peut par exemple chercher quelle entité rejouée a le voisinage topologique le plus proche de l'entité initiale, c'est-à-dire, par exemple, laquelle a le plus d'entités voisines en commun avec l'entité initiale.

L'appariement global consiste à comparer deux ensembles, l'un composé d'entités initiales et l'autre d'entités rejouées, afin de les mettre en correspondance. C'est la méthode utilisée par Kripac [Kri95], par exemple. Nous avons expliqué à la section 2.4.2.2 que ce dernier, pour apparier une face, remonte son arbre d'historique jusqu'à retrouver une face invariante, puis récupère les feuilles issues de cette face au jeu initial et au rejeu, qui

forment alors les deux ensembles exploités pour la mise en correspondance.

La méthode locale a l'avantage de comparer moins d'éléments et donc de nécessiter, *a priori*, moins de temps et de puissance de calcul. La méthode globale, elle, permet *a priori* un appariement plus stable et plus fiable, mais au prix de temps de calcul plus importants.

Après chaque opération, pour chaque nom persistant, notre méthode permet de choisir parmi les brins modifiés par la règle de l'opération, lequel sera le support de l'orbite feuille d'appariement de l'entité initiale. Notre méthode est donc locale. Cependant, grâce à l'identification précise permise par les règles de transformations de graphes, elle est également stable et fiable.

5.2 Taille de la structure de données

Réaliser une comparaison du temps d'exécution des différentes méthodes étudiées serait ardu voire impossible, car notre algorithme est actuellement 1) purement théorique, 2) les articles traitant des autres méthodes ne détaillent pas toujours comment réaliser le rejeu grâce aux systèmes de nommage qu'ils présentent. Cependant, nous pouvons comparer les tailles des structures de données de différents modèles.

Si l'on ne compte pas les journaux de bord des règles, qui sont plus liés aux règles Jerboa pré-existantes qu'au rejeu, notre système de nommage compte trois éléments :

— les noms persistants, une suite d'identifiants persistants ;
— les journaux d'historique, basés sur les journaux de bord et les identifiants persistants ;
— les arbres d'appariement, construits grâce aux journaux d'historique.

Pour notre comparatif, nous allons étudier les méthodes de Kripac, Capoyleas, Wu, et Baba Ali. Nous allons traiter un extrait de notre exemple fil rouge, présenté à la figure 5.1 pour calculer la taille mémoire utilisée par chaque méthode. Afin de pouvoir appliquer les autres méthodes, valables uniquement en dimension 3, nous modifions cependant notre exemple fil rouge 2D original pour traiter à la place un cube.

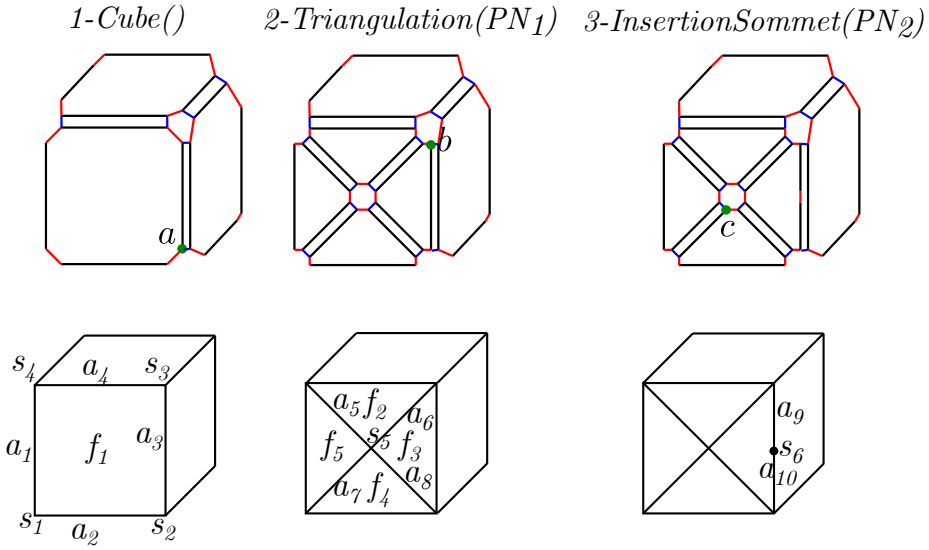


Figure 5.1 : Fil rouge du calcul des tailles de noms persistants

5.2.1 Calcul des tailles des noms persistants

5.2.1.1 Notre méthode

Dans notre méthode, chaque nom est composé d'autant d'identifiants que nécessaire pour caractériser toutes les opérations ayant affecté l'entité qu'il caractérise. Si m est le nombre d'opérations de notre spécification paramétrique, le nom persistant est composé d'au plus $m-1$ identifiants persistants (la m^{e} opération utilisant le nom), chacun pouvant être respectivement constitué d'au plus $1, 2, ..., (m-1)$ éléments. Le nom persistant est donc au maximum constitué de $1+2+...+(m-1)+1 = m*(m-1)/2+1$ éléments, $+1$ pour l'élément type d'orbite. En effet, le pire cas envisageable est celui où le premier identifiant ne tiendrait compte que d'une opération, et où chaque identifiant supplémentaire comprendrait toutes les opérations déjà présentes dans les autres identifiants plus une nouvelle opération. Soit $m = 4$, le plus long nom persistant possible pourrait être, par exemple, $\{\{1-n_0\}\{1-n_0;3-n_3\}\{1-n_0;2-n_2;3-n_3\}\}.TypeOrbite$.

Cependant, ce chiffre est le nombre maximum théorique d'éléments, le nombre réel est bien inférieur. En effet, on récupère un premier identifiant, puis, s'il manque des opérations, on ajoute autant d'identifiants que nécessaire pour obtenir toutes les opérations ayant affecté l'entité. Ainsi, si par exemple il manquait une seule opération dans l'un des identifiants, il en faudrait un deuxième, dont la taille maximale serait $m-1$. La taille

finale serait alors $(m-2)+(m-1)+1 = 2*(m-1)$. Si par exemple, dans un autre cas de figure, tous les identifiants ne rendaient compte que d'une seule opération, le nom ne comprendrait alors que $(m-1)+1$ éléments.

Pour le calcul, nous prendrons la convention suivante : un type d'orbite en dimension 3 est codé sur 4 bits, un pour chaque booléen indiquant si la dimension est utilisée ou pas. Par exemple, une orbite $\langle 0,1,2,3\rangle$ serait représentée par 1111, $\langle 0,2,3\rangle$ par 1011. De même, nous utiliserons une taille de 1 octet pour les différents entiers, en partant du principe qu'il y a au plus 255 opérations dans la spécification, et 255 nœuds utilisés au maximum dans une règle.

Sur notre exemple fil rouge, il y a deux noms persistants, $PN_1.\{\{1-n_4\}\}.\langle 0,1,3\rangle$ et $PN_2 = \{\{1-n_3; 2-n_0\}\}.\langle 0,2,3\rangle$. PN_1 fait donc $1o+1o+4b = 2o+4b$, et PN_2 fait $2*(1+1)o+4b = 4o+4b$, soit un total de 7 octets. Pour information, et même s'il ne fait pas partie de ce fil rouge, notre nom persistant le plus conséquent, PN_5, ferait $(3*2+2*2+3*2)o+3b$, soit 16 octets et 3 bits en 2D, $(3*2+2*2+3*2)o+4b$ soit 16 octets et 4 bits en 3D. Ici, on ne compte pas la taille de chaque identifiant persistant, car, bien qu'ils soient mis à jour à chaque opération, ces informations ne sont pas nécessaires à l'appariement et sont donc supprimées une fois le jeu initial terminé, pour ne garder que les noms persistants.

5.2.1.2 Kripac

Chez Kripac, le nommage est réalisé selon la formule : pour les faces $FaceId(f) = [stepID, faceIndex, surfaceType]$, $EdgeID(e) = [adjFaceIds, endFaceIds_{0,1}, edgeIntersCode]$ pour les arêtes, et $VertexId(v) = [adjFaceIds, vertexIntersCode]$ pour les sommets. Kripac nomme tous les éléments. FaceID est composé de trois éléments, deux entiers (l'ID de l'opération et celui de la face à sa création), et un élément permettant de caractériser le type de surface. La taille d'un FaceID est donc *a priori* assez petite. Cependant, les arêtes et sommets sont nommés en utilisant, entre autres, toutes leurs faces incidentes. Ajouté au fait que Kripac nomme toutes les entités, on peut s'attendre à ce que la taille de la partie nommage augmente plus vite.

Sur notre exemple, à l'étape 1, la face centrale f_1 est nommée $[1,1,plane]$, soit une taille de $(1+1+1)o$, soit 3 octets, de même pour les 5 autres faces. À cela s'ajoutent les 12 arêtes, nommées $[ID(f_{incidentes}), ID(f_{extremités}), int]$, chacune ayant deux faces incidentes

et deux faces à ses extrémités, et qui font donc chacune $(2*3+2*3+1)o = 13$ octets. Enfin, 8 sommets sont également créés, nommés $[ID(f_{incidentes}), int]$, chacun ayant trois faces incidentes de $(3*3+1)o = 10$ octets chacun. À la première étape, on a donc 26 noms pour un total de 254 octets.

Il faut ajouter à ce total les faces, arêtes et sommet construits à l'étape 2, $f_2, ...f_5$, $a_5, ...a_8$ et s_5, qui font 3 octets pour chaque face, $2*3+2*3+2*3+1 = 19$ octets pour chaque arête, et $4*3+1 = 13$ octets pour le sommet. On passe donc à 355 octets. Enfin, à l'étape 3, on crée 1 nouveau sommet et 2 arêtes, le sommet de 7 octets, et les arêtes, de 10 octets chacune, pour un total de 382 octets. Nous comparons l'ensemble des tailles de structures de nommages et d'appariement à la section 5.2.3.

5.2.1.3 Capoyleas

Pour la méthode de Capoyleas, les entités sont nommées en fonction de l'entité de base qui a permis leur construction, $e(v)$ par exemple pour une arête e créée par extrusion d'un sommet v, et les entités de base sont nommées grâce à l'identifiant de l'opération les ayant créées et grâce à leur identifiant. Par exemple, l'arête 2 créée à l'étape 1 sera nommée A :1.2

La méthode de Capoyleas fonctionne sur les opérations créant des volumes à partir d'esquisses, ce qui n'est pas notre cas ici. Imaginons pour ce cas-ci que notre cube soit en réalité issu de l'extrusion de la face f_1, afin de pouvoir appliquer la méthode. La taille de $f_1, a_1...a_4, s_1...s_4$ est de $2o+2b$, un octet pour le numéro d'opération, un pour l'identifiant de l'entité, deux bits permettant de représenter le type d'entité en 3D. On a alors 20 octets et 2 bits de données. À cela, on ajoute les 8 arêtes, 5 faces, 4 sommets et 1 volume. On a à nouveau besoin de l'information du type, 2 bits, et d'un numéro pour savoir en combientième a été créée l'entité, ce qui donne $1o+2b+(2o+2b) = 3o+4b$.

À l'étape 2, les entités ne sont pas issues d'une esquisse. Les faces scindées héritent donc du nom de leur face mère, f_1, soit une taille de $2o+2b$. Le sommet est nommé grâce à ses faces adjacentes, un booléen indiquant l'orientation, et une information sur ses caractéristiques, ce qui donne une taille de $4*(2o+2b)+1b+(1o+4*(1o+1o)) = 18o+1b$. Chaque arête est nommée grâce à toutes ses faces adjacentes, une orientation locale elle-même composée des sommets adjacents à l'arête et d'un booléen, et une information sur ses caractéristiques. On a donc une taille pour chaque arrête de $2*(2o+2b)+(2o+2b+$

$18o + 1b + 1b) + (1o + 2 * (1o + 1o)) = 30o$.

À l'étape 3, on scinde l'arête, les deux arêtes résultantes auront donc une taille de $2o+2b$, et le sommet résultant, de $2 * (2o + 2b) + 1b + (1o + 2 * (1o + 1o)) = 9o + 5b$.

Capoyleas les utilise afin de résoudre les ambiguïtés dans le cas d'entités ayant un nom identique (à cause d'une scission par exemple, les entités filles héritant toutes du nom de l'entité mère). Ce système de résolution repose sur les distances de voisinages des entités. On prendra tout d'abord ses voisins directs, puis les voisins de ces voisins... jusqu'à trouver un nom permettant de discriminer totalement l'entité. Il est à noter cependant que, pour optimiser cet ensemble de noms, Capoyleas ne considère que les noms discriminants de chaque distance de voisinage, ou, quand il n'y en a pas, un seul des noms présents dans la distance de voisinage (voir section 2.4.2.2).

À la première étape, chacune des faces peut être caractérisée par elle-même uniquement, sans avoir recours à son voisinage. En effet, il n'y a pas deux faces ayant le même nom $(f_1, f(a_1)...)$. f_1, par exemple, sera caractérisée par $[0, f_1, 1]$, soit une taille de $[1o+2o+2b+1o] = 4o + 2b$. De manière générale, pour toutes les entités, la taille de leur discriminant sera égale à la taille de leur nom plus 2 octets, pour un total à l'étape 1 de 135 octets et 2 bits.

À l'étape 2, les faces ont toutes le même nom, hérité de leur mère. il faudra donc aller chercher une des arêtes de leur voisinage pour pouvoir les distinguer. Pour f_5 par exemple, on aura $[0, f_1, 1][1, a_1, 1]$, soit une taille de $[1o+2o+2b+1o]+[1o+2o+2b+1o] = 8o+4b$, de même pour les 3 autres faces. Pour les arêtes, on peut également les caractériser grâce au sommet de leur premier voisinage. Chacune a donc une taille de $[1o + 30o + 1o] + [1o + 2o + 2b + 1o] = 36o + 2b$. Enfin, le sommet peut être caractérisé sans avoir recours à son voisinage, soit une taille de $[1o + 18o + 1b + 1o] = 20o + 1b$, pour un total de 343 octets et 1 bits.

À l'étape 3, les deux arêtes peuvent être caractérisées par le sommet de leur premier voisinage, pour une taille de $[1o + 2o + 2b + 1o] + [1o + 2o + 2b + 1o] = 8o + 4b$ chacune, et le sommet par lui seul, pour une taille de $[1o + 9o + 5b + 1o] + = 11o + 5b$, soit une taille totale de 28 octets et 5 bits.

La taille total du nommage à l'étape 1 est donc de 507 octets

5.2.1.4 Wu

Comme celle de Capoyleas, la méthode de Wu fonctionne sur les opérations créant des volumes à partir d'esquisses. Imaginons à nouveau que notre cube soit en réalité issu de l'extrusion de la face f_1, afin de pouvoir appliquer la méthode. Il nomme les faces issues de ces extrusions :

$$ON(F) = [FeatID, FeatID_p, ID_{element}, FeatID_{Path}, ID_{trajectory}]$$

Les faces issues d'une scission héritent du nom de leur mère, mais une information géométrique PSI est calculée puis ajoutée à ce nom afin de distinguer ces faces entre elles. Le nom devient alors :

$RN(F) = [ON(f), PSI]$ avec $PSI = ON(f); Seq; Totle$ Seq et Totle étant deux entiers [WZZ01].

Pour chacune de nos faces créées à l'étape 1, on obtient donc une taille de $5 + 7 = 12$ octets. Les 4 faces créées à l'étape 2 ont également un nom de 12 octets, pour un total de 120 octets.

Pour les arêtes, le nom est :

$$RN(E) = [RN(F_1), RN(F_2), PSI]$$

La taille de chacune des 18 arêtes est donc de $7 + 7 + 7 = 21$ octets, car PSI est basée sur le ON d'une seule face adjacente. On obtient un total de 378 octets.

Enfin, les sommets sont nommés

$$RN(V) = [RN(E_1), RN(E_2), PSI]$$

Chacun des 10 sommets a donc une taille de $21 + 21 + 7 = 49$ octets, car PSI est basée sur le ON d'une seule face adjacente. On obtient un total de 490 octets.

Avec cette méthode, on obtient une taille totale de 988 octets pour les noms persistants du fil rouge.

5.2.1.5 Baba Ali

Chez Baba Ali, on a quatre formes de nom :

[numéro d'étape ; numéro de l'entité] pour les arêtes invariantes

[numéro d'étape ; ensemble des faces invariantes incidentes ;
ensemble des entités et agrégats invariants incidents au premier sommet de l'arête ;
ensemble des entités et agrégats invariants incidents de l'arête] pour les arêtes contingentes

[numéro d'étape ; numéro de l'agrégat] pour les agrégats invariants

[numéro d'étape ; ancêtre invariant ; arêtes de l'agrégat] pour les agrégats contingents

À l'étape 1, on ne crée que des entités invariantes. Elles ont donc une taille de $1 + 1 = 2$ octets, pour un total de 52 octets, en partant du principe que les sommets invariants sont nommés numéro d'opération / numéro d'entité.

À l'étape 2, on a la création de 4 faces contingentes, 4 arêtes contingentes, et 1 sommet invariant. Le sommet aura comme précédemment une taille de 2 octets. Les 4 arêtes auront une taille de $1 + 0 + 4 * 2 + 0 = 9$ octets, l'arête étant incidente à des faces contingentes, ayant 2 arêtes et 2 faces invariantes incidentes à son premier sommet, et n'ayant que des entités contingentes incidentes à son deuxième sommet. La taille de chacune des face est de $1 + 2 + (2 + 9 + 9) = 23$ octets, pour un total de 130 octets.

À l'étape 3, on crée 2 arêtes contingentes et 1 sommet invariant. En reprenant les mêmes méthodes que précédemment, on a donc une taille de 2 octets pour le sommet, et de $1 + 2 + 4 * 2 + 4 * 2 = 19$ octets pour chaque face, pour un total de 40 octets, et une taille totale de 222 octets.

	Notre méthode	Kripac	Capoyleas	Wu	Baba Ali
Noms :	7 octets	382 octets	507 octets	988 octets	222 octets

Table 5.1 : Taille totale du nommage

5.2.2 Appariement

5.2.2.1 Notre méthode

Dans notre méthode, l'appariement se réalise avec deux éléments : les journaux d'historique, et les arbres d'appariement.

Le journal d'historique est composé d'une succession d'ensembles numéro d'opération / numéro de nœud, d'un type de modification, d'une taille de 3 bits (pour les 6 types de modification identifiés) et d'un entier représentant le nœud à l'origine de la branche. La taille mémoire d'un journal de bord est donc celle de l'identifiant persistant qu'il caractérise, auquel on ajoute le type de modification, 3 bits, le numéro du nœud à l'origine de la chaîne (1 octet), et un type d'orbite (4 bits en 3D) pour chaque branche du journal.

Dans notre exemple, nous avons deux journaux, présentés à la figure 5.2. Les autres méthodes ne présentant pas l'édition, nous allons étudier le cas d'un rejeu identique.

La taille du premier journal sera donc de $(2 * 1 + 1)o + 3b + 4b = 3o + 7b$, et celle du deuxième sera de $[(2 * 1 + 1)o + 3b + (4b)] + [(2 * 1 + 1)o + 3b + (4b)] = 7o + 6b$, soit un total de 11 octets et 5 bits.

Il faut maintenant prendre en compte les arbres d'appariement. Leurs branches sont constituées du type d'édition, codé sur 2 bits, et du type de modification, sur 3 bits, et d'un brin associé à un type d'orbite, brin qui sera désigné par un identifiant sur 2 octets au lieu de 1 octet, afin d'avoir une comparaison plus juste avec les autres méthodes. En effet, une face carré contient 8 brins. Cependant, elle partage ces brins avec une entité de chaque dimension. En dimension 3, un brin fait donc partie de 4 entités, d'où une taille de $8/4 = 2$ octets. Il faut ajouter comme racine la taille de l'identifiant persistant à l'origine de la branche.

Pour le premier nom persistant, on a donc une taille de $2o+4b+(2o+3b+2b)+4b = 5o+5b$ et pour le deuxième $4o + 4b + (2o + 3b + 2b) + 4b + (2o + 3b + 2b) + 4b) = 10o + 6b$, pour un total de 16 octets 3 bits, et un total pour l'appariement et l'historique de 28 octets.

5.2.2.2 Kripac

Chez Kripac, l'appariement se fait grâce à un arbre d'historique. Celui-ci est composé des noms persistants précédemment créés, liés par un indicateur du type de changement, pour lequel nous prendrons comme précédemment une taille de 3 bits. De plus, cet arbre

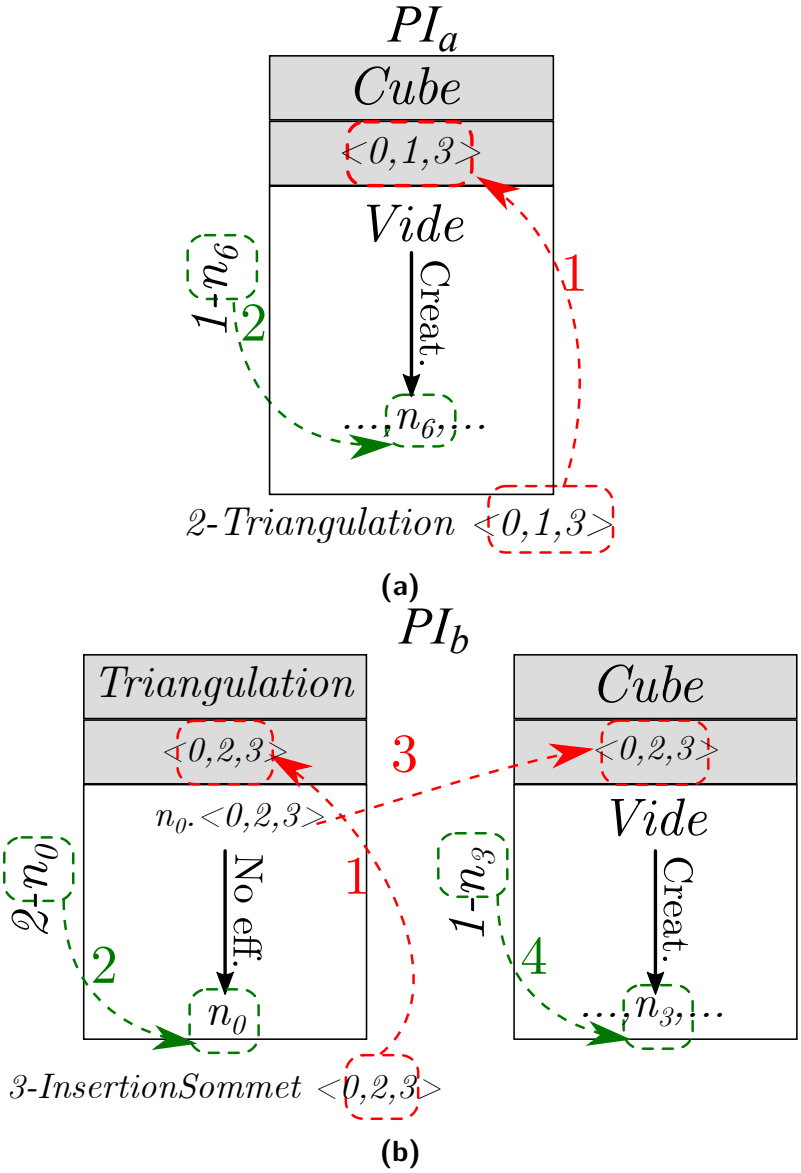


Figure 5.2 : Journaux d'historique 3D

Entités de départ	Entités d'arrivée	Type d'événement
id_F	$\{id_{F_1}, id_{F_2}\}$	scission
id_{A_1}	$\{id_{A_{1.1}}, id_{A_{1.2}}\}$	scission
id_{A_2}	$\{id_{A_{2.1}}, id_{A_{2.2}}\}$	scission
...	...	...
$\{\}$	$\{id_{S_1}\}$	création
$\{\}$	$\{id_{S_2}\}$	création

Figure 5.3 : Suivi d'historique chez Baba Ali, extrait de [BA10]

ne comprend que les faces.

Dans notre exemple, nous avons 5 arbres qui n'évoluent pas, ceux des faces non modifiées du cube. Ceux-ci font donc chacun une taille de $3o + 3b$. Nous avons également un arbre permettant d'indiquer que la face f_1 a été créée, puis scindée en 4 faces : f_2, f_3, f_4 et f_5, chaque branche portant un label indiquant la scission, ce qui donne donc un arbre de $3o + 3b + 4 * (3o + 3b) = 16o + 7b$, pour un total de 33 octets et 6 bits.

Concernant l'appariement des arêtes et des sommets, celui-ci se réalise grâce à l'appariement de leurs faces voisines, il n'y a donc pas *a priori* plus de données à stocker.

5.2.2.3 Capoyleas

Chez Capyleas, l'historique et l'appariement des entités ne sont pas abordés.

5.2.2.4 Wu

Chez Wu, l'historique consiste à propager le nom ON d'une entité aux enfants de celle-ci.

A l'étape 1, il n'y a donc pas d'historique. A l'étape 2, la face scindée hérite du nom de sa mère, d'une taille de 12 octets, et à l'étape 3, les deux arêtes scindées héritent chacune du nom de leur mère, de 21 octets, soit un total de 54 octets.

5.2.2.5 Baba Ali

Chez Baba Ali, le suivi de l'historique se fait *via* un tableau, comme celui présenté à la figure 5.3. Sa taille est donc basée sur celles des noms, et sur celle du type de modification 3 bits.

Dans notre exemple, après la création du cube, on a la création de 6 faces, 12 arêtes et 8 sommets, chacune de ces entités prenant dans le tableau la taille de $0 + 2o + 3b = 2o + 3b$, pour un total de 61 octets et 6 bits. À l'étape 2, on crée 4 arêtes et 1 sommet, et on scinde une face en deux, ce qui ajoute $0 + 2o + 3b = 2o + 3b$ pour le sommet, $0 + 9o + 3b = 9o + 3b$ pour chaque arête, et $1 * 2o + 4 * 23o + 3b = 94o + 3b$ pour les entités scindées, soit un total de 133 octets et 6 bits. Enfin, à l'étape 3, on crée un sommet et on scinde une arête, ce qui ajoute $0 + 2o + 3b = 2o + 3b$ pour le sommet et $2o + (2 * 19)o + 3b = 40o + 3b$ pour la scission. On a donc un total de 236 octets et 2 bits.

Ces tailles mémoires dues a l'appariement sont regroupées dans le tableau 5.2.

	Notre méthode	Kripac	Capoyleas	Wu	Baba Ali
Appariement :	28 octets	33 octets et 6 bits	-	54 octets	236 octets et 2 bits

Table 5.2 : Taille totale des mécanismes d'appariement

5.2.3 Résumé

	Notre méthode	Kripac	Capoyleas	Wu	Baba Ali
Noms :	7 octets	382 octets	507 octets	988 octets	222 octets
Appariement :	28 octets	33 octets et 6 bits	-	54 octets	236 octets et 2 bits
Total :	35 octets	415 octets et 6 bits	507 octets	1042 octets	458 octets et 2 bits

Table 5.3 : Résumé de la taille totale du nommage et des mécanismes d'appariement

Le tableau 5.3 permet de mettre en parallèle tous ces résultats. Notre méthode semble efficace en terme de taille mémoire, grâce au fait que l'on ne nomme et que l'on ne suive

que les entités réellement utilisées pour le rejeu. En effet, la partie nommage utilise 55 fois (resp. 70, resp. 140, resp. 30) moins de place que celle développée par Kripac (resp. Capoyleas, resp. Wu, resp. Baba Ali).

La partie appariement, elle, est au coude à coude avec celles de Kripac et de Wu, et utilise néanmoins 8 fois moins de place que celle de Baba Ali. Cela est dû au fait que Kripac n'apparie que les faces, et réalise ensuite l'appariement des autres entités *via* celles-ci, là où les deux autres méthodes apparient toutes les entités directement. Wu, quant à lui, ne détaille pas la partie appariement de sa méthode, mais seulement la partie historique.

5.3 Résumé

En résumé, nous proposons une méthode homogène et générale de nommage des entités, qui permet l'édition de spécifications paramétrique. De plus, le fait de nommer uniquement les entités utilisées au rejeu a permis d'optimiser la taille du nommage. La taille de l'appariement est elle aussi inférieure à celle des autres méthodes, malgré le fait qu'il faille stocker les informations d'historique pour toutes les entités nommées. Le tableau 5.4 permet de résumer les différents points abordés dans ce chapitre.

	Notre méthode	Kripac	Capoyelas	Wu	Baba Ali
Édition :	Oui	Non	Non	Non	Non
Homogène :	Oui	Non	Non	Non	Non
Général :	Oui	Non	Non	Non	Oui
Nommage basé sur :	Les brins	L'entité de dimension $n-1$	L'entité de dimension $n-1$	L'entité de dimension $n-1$	Les arêtes
Nomme	Les entités utilisées au rejeu	Toutes les entités	Toutes les entités	Toutes les entités	Toutes les entités
Taille des noms dans l'exemple :	7 octets	382 octets	507 octets	988 octets	222 octets
Taille des mécanismes d'appariement dans l'exemple :	28 octets	33 octets et 6 bits	-	54 octets	236 octets et 2 bits

Table 5.4 : Résumé du système de nommage persistant

When all is said and done

ABBA

6 Conclusion générale

Sommaire

Une modélisation paramétrique, de part son utilisation pour réaliser un rejeu, nécessite l'utilisation d'un système de nommage à la fois non ambigu, afin de permettre de discriminer les entités, mais également contenant suffisamment d'informations pour permettre d'apparier ces dernières. En effet, le rejeu d'une spécification paramétrique peut entraîner diverses modifications (scission, fusion, suppression...) des entités, et donc rendre

impossible le suivi des références contenues dans la spécification initiale.

Dans le cadre de notre travail, nous nous sommes intéressés au problème suivant : concevoir un système de nommage et d'appariement suffisamment robuste pour pouvoir éditer une spécification paramétrique, problème peu abordé dans la littérature, qui se penche plutôt sur l'édition de paramètres.

Dans un premier temps, nous allons donc aborder le travail réalisé pour résoudre le problème du nommage persistant et de l'édition de spécification paramétrique, avant de proposer quelques perspectives dans une deuxième partie.

6.1 Travail réalisé

6.1.1 Étude de l'existant

Dans notre état de l'art, nous avons montré certaines limitations et difficultés présentes dans les méthodes existantes de nommage persistant.

Pour cela, nous avons commencé par un bref historique de la modélisation, avant d'enchaîner sur les différentes manières possibles de réaliser une modélisation paramétrique.

Celle-ci étant composée, entre autres, d'une représentation géométrique, nous avons fait un rapide inventaire des modèles géométriques existants, en nous attardant sur celui que nous voulions utiliser, le modèle topologique des cartes généralisées (G-Cartes). En effet, l'entité de base de ces G-Cartes, le brin, permet de désigner les entités manipulées dans la modélisation paramétrique et semblait donc une bonne base pour un système de nommage général et homogène.

Nous nous sommes ensuite penchés sur le problème de la nomination persistante et sur les diverses solutions qui y ont déjà été apportées. Nous avons identifié d'une part certains concepts communs que nous allions également utiliser, comme la notion d'historique dans le nom. D'autre part, nous avons également mis en avant certaines limitations que nous voulions supprimer de notre système, comme la résolution d'ambiguïtés basée sur la géométrie, peu fiable, ce qui a renforcé l'idée d'utiliser un modèle topologique, ou encore, le non-traitement de l'édition de spécification.

Enfin, il a fallu déterminer une manière de concevoir nos spécifications paramétriques. Les G-Cartes étant des graphes, nous nous sommes tournés vers les règles de transforma-

tions de graphes, et plus précisément, vers les règles Jerboa, conçues pour transformer des G-Cartes, et donc parfaites pour servir de base à nos opérations.

6.1.2 Apports

Afin de pouvoir réaliser notre système de nommage persistant permettant l'édition de spécifications paramétriques, nous avons procédé en deux étapes.

Nous avons ainsi étudié :

— Tout d'abord, la création d'un identifiant persistant pour chaque brin de la G-Carte, construit grâce aux règles Jerboa.

— Ensuite, la création du nom persistant, permettant de caractériser toutes les entités d'une G-Carte, construit grâce aux identifiants persistants des brins de l'orbite désignée, et au type de celle-ci. Les entités d'une G-Carte pouvant toutes être désignées par la paire constituée d'un de leurs brins et de leur type d'orbite, ce système de nommage est bien général et homogène.

  De plus, afin d'optimiser l'espace de stockage, le nom est uniquement composé du nombre minimal d'identifiants nécessaires à la caractérisation de l'ensemble des règles ayant impacté l'entité, et non pas des identifiants de tous les brins de l'orbite, et on ne crée que les noms des entités utilisées par la spécification paramétrique.

Il a ensuite fallu compléter ce modèle afin de permettre l'édition d'une spécification paramétrique, en utilisant :

— Les journaux de bord, permettant de suivre les modifications apportées par la règle aux entités, construits par l'utilisateur en même temps que la règle au sein du modeleur.

— Les journaux d'historique, permettant de suivre toutes les opérations ayant affecté les entités nommées au jeu initial, construits grâce aux noms persistants et à des extraits des journaux de bord des règles que référencent ces noms.

— Les arbres d'appariement, construits au rejeu et permettant à chaque opération du rejeu, de savoir quelle(s) entité(s) est (sont) appariée(s) à chaque entité désignée par un nom persistant, construits grâce aux journaux d'historique.

Ces travaux ont également fait l'objet de deux articles, pour les conférences WSCG [MCSDG18] et Computer-Aided Design and Applications [CMS18]. Les références à ce dernier article sont celles du proceeding, le journal Computer-Aided Design and Applica-

tions étant encore en cours de mise en ligne à l'heure où nous écrivons ces lignes.

6.1.3 Comparaison à l'existant

Nous avons donc pu mettre en place non seulement un système de nommage persistant général et homogène, mais également un ensemble de mécanismes permettant d'éditer une spécification paramétrique (ajouter, supprimer ou déplacer une opération), là où les méthodes existantes ne traitent que l'édition de paramètres d'opérations, et non de spécifications.

De plus, notre approche utilise les règles de transformation de graphes et permet donc, d'une part, d'utiliser ces dernières pour créer un système de nommage, mais également d'autre part, de réaliser des spécifications paramétriques à base de règles, ce qui est également un domaine peu abordé dans la littérature.

Nous avons également pu étudier l'adéquation de notre système de nommage avec les concepts communs des systèmes de nommage existants. Nous retrouvons notamment dans notre système la notion d'architecture à trois couches, et celle d'historique dans la nomination, grâce à nos noms tenant compte des différentes opérations.

Enfin, nous avons pu confronter notre système de nommage et d'appariement à divers systèmes existants en comparant la taille mémoire nécessaire à son exécution sur un exemple. Notre système semble avoir de bonnes performances, car il a nécessité moins d'espace mémoire à la fois pour le nommage et pour l'appariement que les autres méthodes. Toutefois, de nouveaux tests seront nécessaires une fois l'algorithme codé, afin de s'assurer de la rapidité de cette méthode, et de son efficacité mémoire sur des exemples plus complexes.

6.2 Perspectives

6.2.1 Scripts de règles

Dans ce manuscrit, nous avons seulement manipulé les règles Jerboa élémentaires, c'est-à-dire celles pouvant être directement décrites dans le modeleur (création d'entités, triangulation, coloration...). Cependant, nous n'avons pas abordé les *scripts de règles*, permettant de réaliser des opérations plus complexes, comme par exemple des opérations booléennes ou plus simplement l'intersection d'une face par une droite. En effet, si l'on considère ce

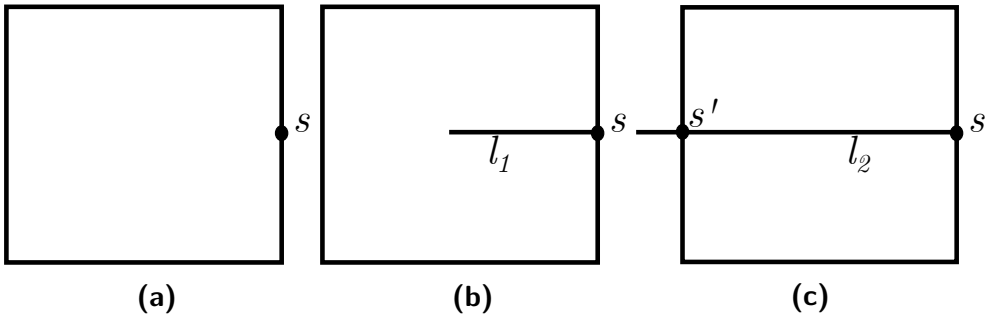


Figure 6.1 : Nécessité d'un script. a : modèle sur lequel est réalisée l'insertion de segment, s le sommet de départ. b : Insertion à partir de s d'un segment de longueur l_1 inférieure à celle du carré. c : Insertion à partir de s d'un segment de longueur l_2 supérieure à celle du carré. On constate alors que dans le premier cas, la face est modifiée, et scindée dans le deuxième. De plus, dans le deuxième cas cas, on doit calculer la position de s' et le créer avant de réaliser l'insertion.

dernier exemple, afin d'insérer une droite (connaissant son équation) dans une face, il faut d'abord calculer les coordonnées des sommets d'intersection, puis joindre ceux-ci entre eux, en faisant appel aux règles élémentaires d'insertion de sommet et d'insertion d'arête. En cas de rejeu, tous les calculs doivent être refaits, et rien ne garantit qu'il y ait autant d'insertions de sommets et d'arêtes qu'au jeu initial. La figure 6.1 illustre la nécessité d'un script sur le cas d'une insertion d'un segment perpendiculaire à un coté d'une face, et d'une longueur définie.

Ces scripts de règles ne sont pas encore disponibles, mais sont développés dans la thèse de Gauthier [Gau18], actuellement en fin de rédaction. Avant de pouvoir y étendre notre algorithme, il faut donc d'abord que les travaux sur les scripts soient testés et validés. Il sera ensuite possible de réaliser des travaux visant à adapter notre système de nommage aux scripts de règles de Gauthier.

6.2.2 Rejeu pour la modélisation historique et archéologique

Enfin, le temps nécessaire au développement de notre algorithme ne nous a malheureusement pas permis de développer la composante archéologique et historique initialement prévue dans la thèse. Une piste à explorer pour réaliser le rejeu historique (à savoir, la possibilité de suivre l'évolution d'un bâtiment au cours du temps) serait d'ajouter aux opérations un paramètre "années", permettant d'appliquer certaines opérations en fonction de l'année désignée par l'utilisateur. On aurait alors par exemple :

— $1 - Domus(\{50, 60, 64, 70\}, 4, 6, 7)$
— $2 - Colonnes(\{60, 64, 70\}, 5, 7)$
— $3 - Atrium(\{60, 70\}, 4, 2, 3.5)$

avec, entre accolades, la liste des années pour lesquelles appliquer l'opération, et ensuite les dimensions des constructions.

Ainsi, pour obtenir par exemple le modèle du bâtiment en l'an 50, on aurait seulement le domus, en 60 et 70, on aurait les trois éléments, et en 64, seulement le domus et les colonnes (en supposant par exemple que l'atrium ait été détruit par l'incendie de Rome). L'une des dates serait celle de la spécification initiale, et les autres seraient des rejeux en éditant la spécification pour ajouter, supprimer ou déplacer des opérations, et changer des paramètres.

Ceci n'est pour l'instant qu'une piste, qu'il faudra donc affiner et valider par une étude et des tests.

6.3 Synthèse

Éditer des spécifications paramétriques est un problème peu abordé dans la littérature, à l'exception de l'édition de paramètres. Nous avons donc voulu, grâce à ces travaux de thèse, créer un système de nommage et d'appariement permettant de pallier ce problème, afin d'avoir plus de souplesse dans le rejeu d'une spécification paramétrique. Malgré cela, des travaux supplémentaires seront à mener afin de l'étendre aux scripts de règles, et donc de réaliser des opérations plus complexes. Cependant, notre modèle semble prometteur en termes d'utilisation, non seulement car il permet l'édition, mais aussi car il nécessite moins de mémoire que les méthodes existantes, et car l'utilisation des règles permet du même coup de réaliser des spécifications paramétriques basées sur celles-ci.

A

Journaux de bord de règles

Sommaire

Dans cette annexe, nous allons présenter, pour chacune des règles utilisées par notre exemple fil rouge, sa forme, un exemple d'application, et son journal de bord.

A.1 Carré

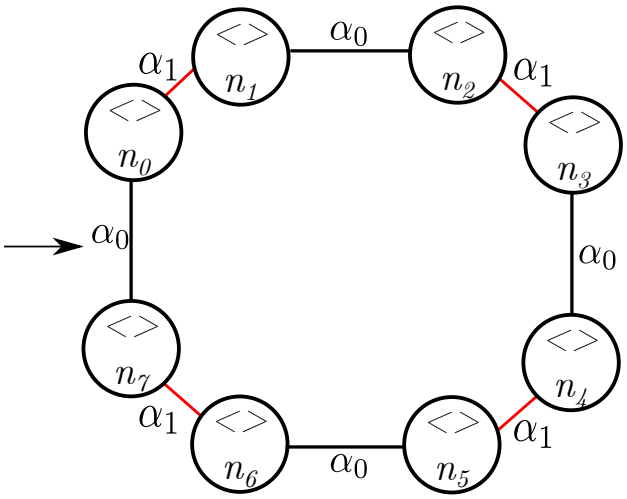


Figure A.1 : Règle de création de carré

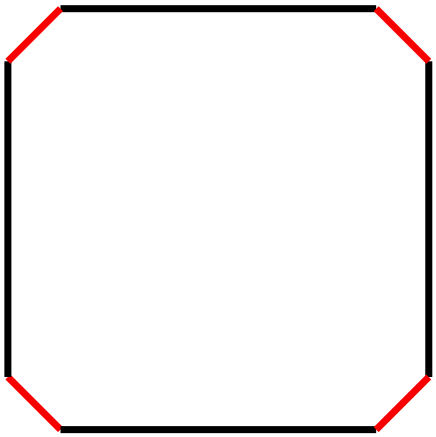


Figure A.2 : Application de la création de carré

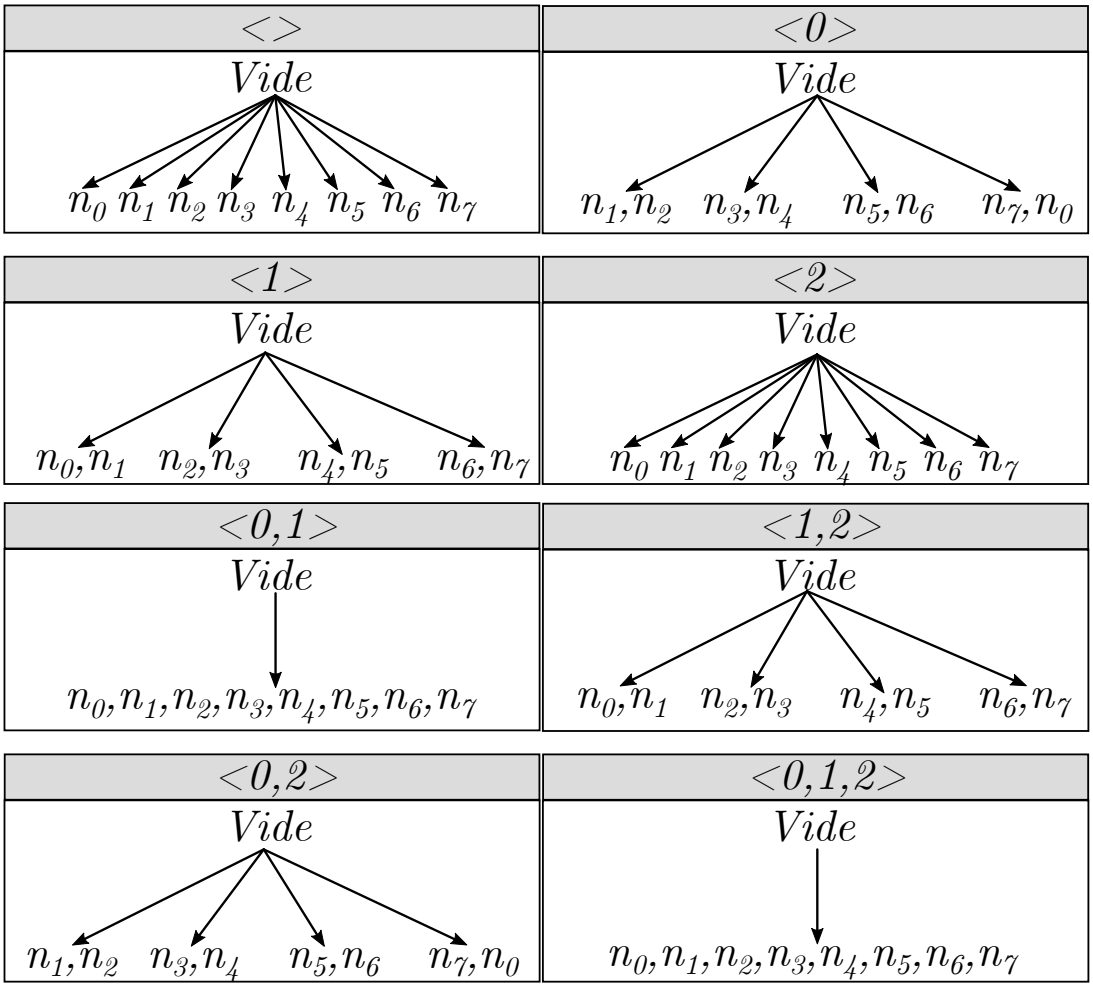


Figure A.3 : Journal de bord de la création de carré. Afin de ne pas surcharger le tableau, le type de modification n'a pas été noté sur les flèches, mais il s'agit systématiquement d'une création.

A.2 Triangulation

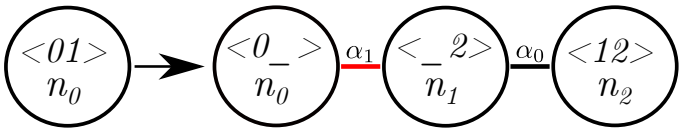


Figure A.4 : Règle de triangulation

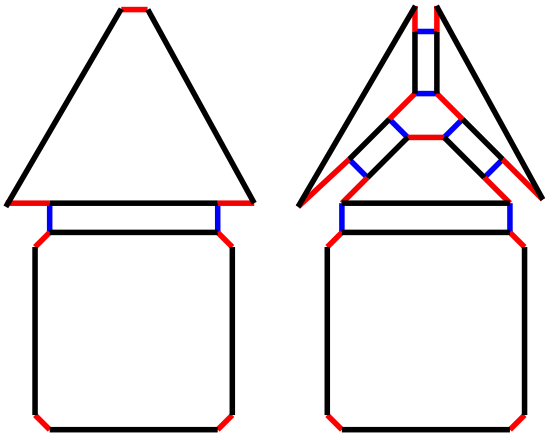


Figure A.5 : Application de la triangulation. A gauche : figure avant la triangulation de la face triangulaire. A droite : résultat de la triangulation de la face triangulaire.

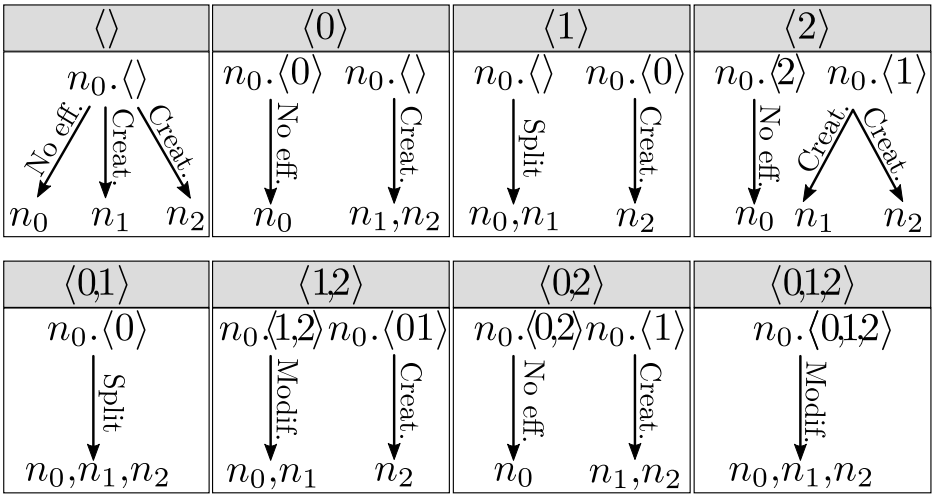


Figure A.6 : Journal de bord de la triangulation

A.3 Insertion de Sommet

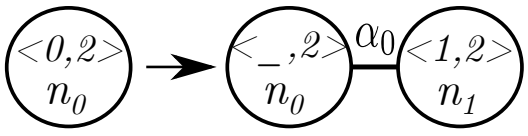


Figure A.7 : Règle d'insertion de sommet

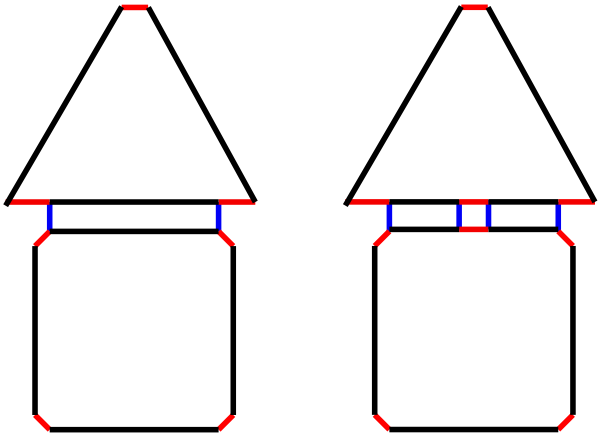


Figure A.8 : Application de l'insertion de sommet. A gauche : modèle avant l'insertion. A droite : modèle après l'insertion sur l'arête séparant le carré du triangle

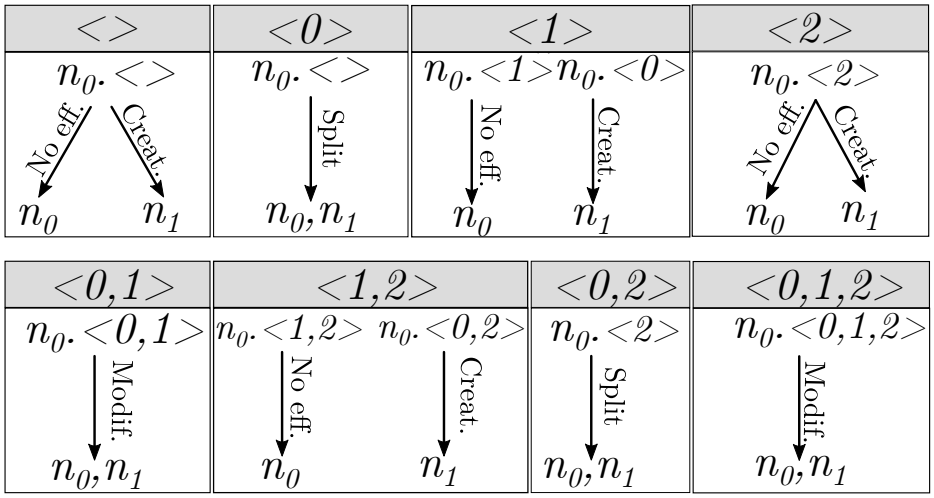


Figure A.9 : Journal de bord de l'insertion de sommet

A.4 Coloration

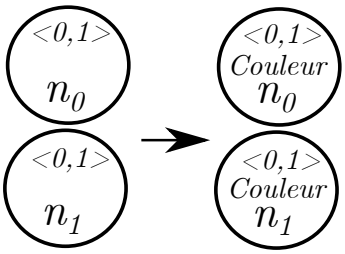


Figure A.10 : Règle de coloration

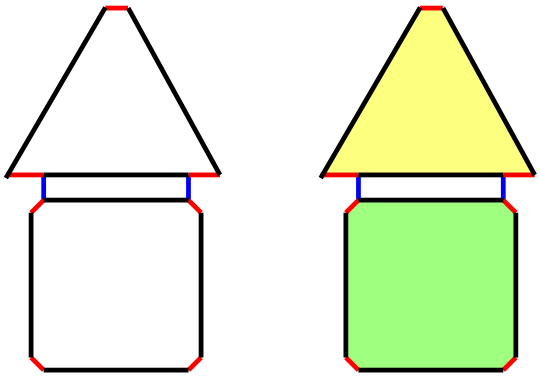


Figure A.11 : Application de la coloration. A gauche : modèle avant l'application. A droite : modèle après l'application de la coloration, en jaune de la face triangulaire, en vert de la face carrée.

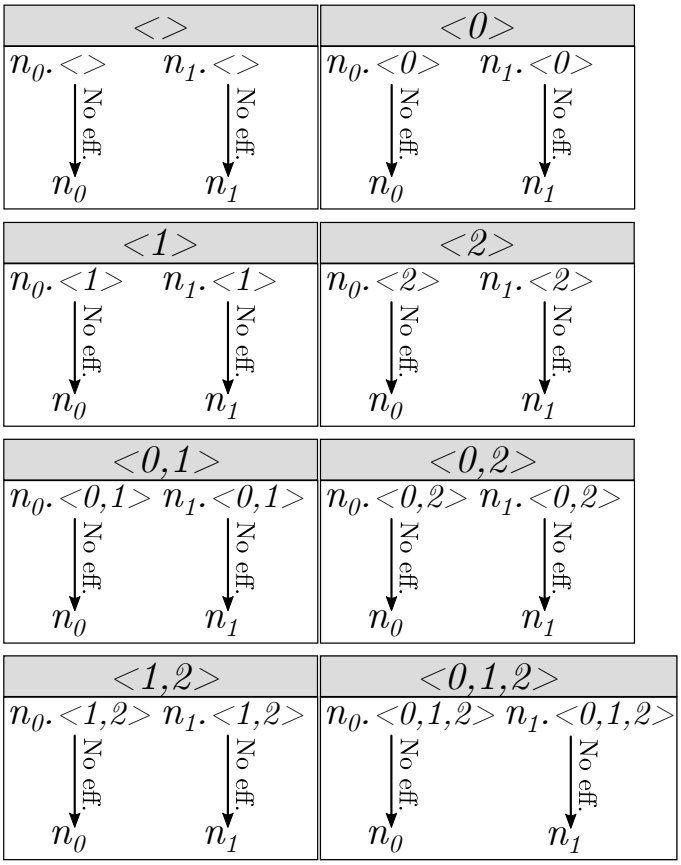


Figure A.12 : Journal de bord de la coloration

B

Journaux d'historique de nos noms persistants

Sommaire

Dans cette annexe, nous rappelons, pour chaque nom persistant, son ou ses journaux d'historique.

B.1 Journaux d'historique initiaux

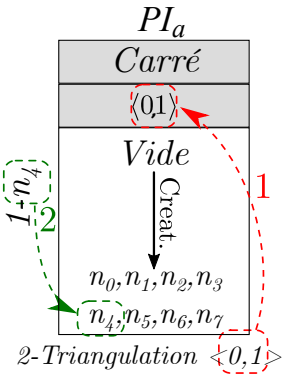


Figure B.1 : Journal d'historique de PI_{1a}

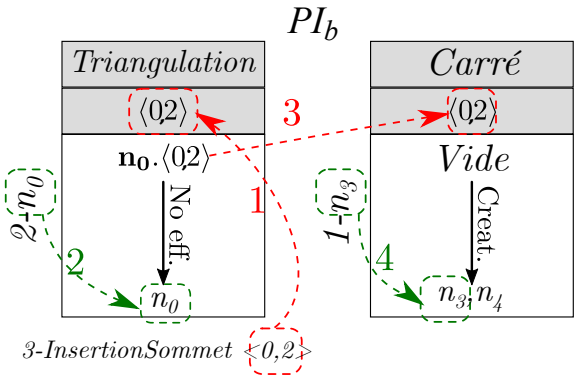


Figure B.2 : Journal d'historique de PI_{2b}

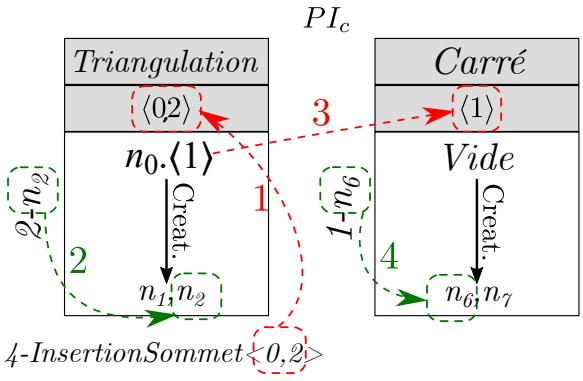


Figure B.3 : Journal d'historique de PI_{3c}

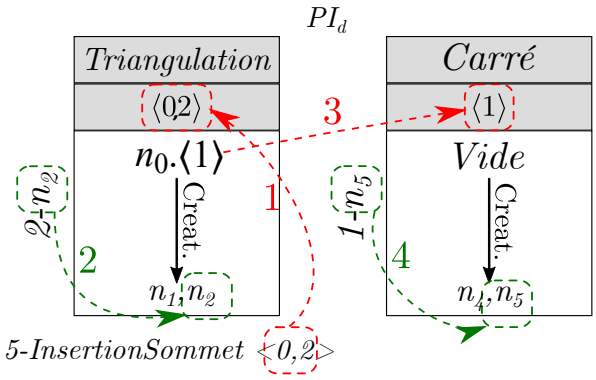


Figure B.4 : Journal d'historique de PI_{4e}

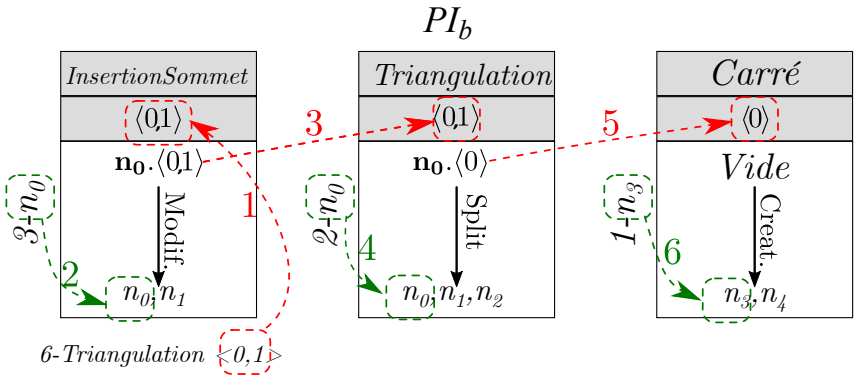


Figure B.5 : Journal d'historique de PI_{5b}

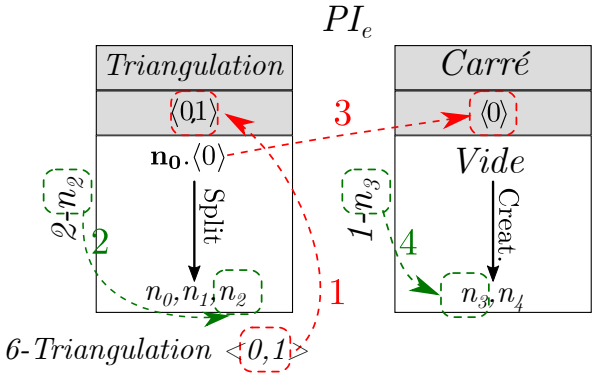


Figure B.6 : Journal d'historique de PI_{5e}

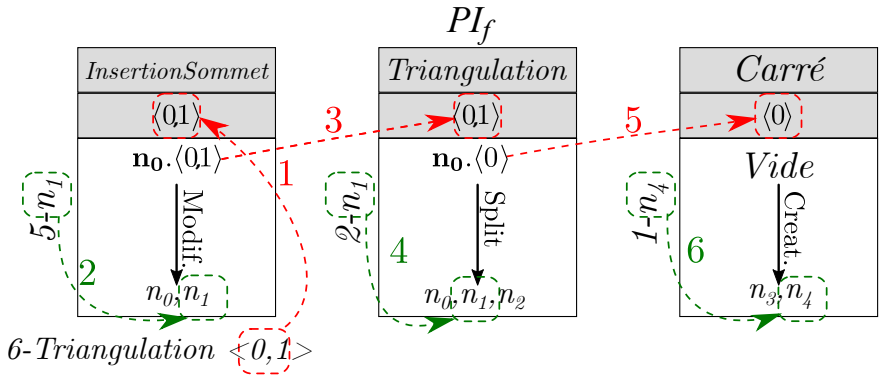


Figure B.7 : Journal d'historique de PI_{5f}

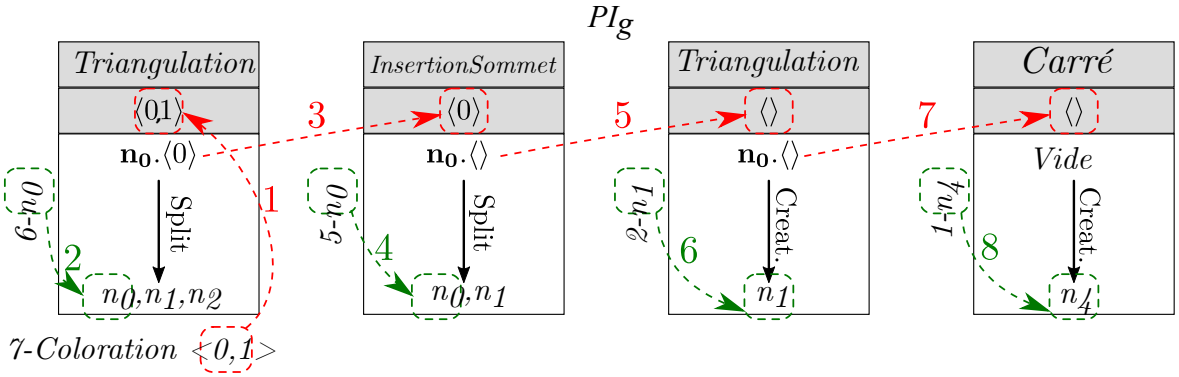


Figure B.8 : Journal d'historique de PI_{6g}

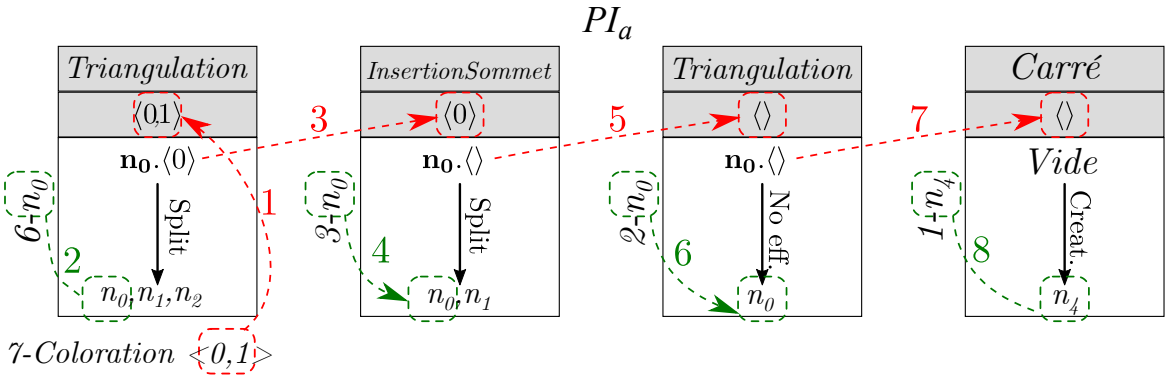


Figure B.9 : Journal d'historique de PI_{7a}

B.2 Journaux d'historique modifiés

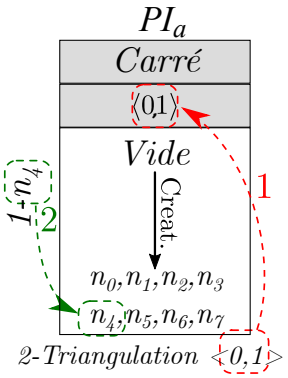


Figure B.10 : Journal d'historique de PI_{1a} - non modifié en raison d'un ajout d'opérations

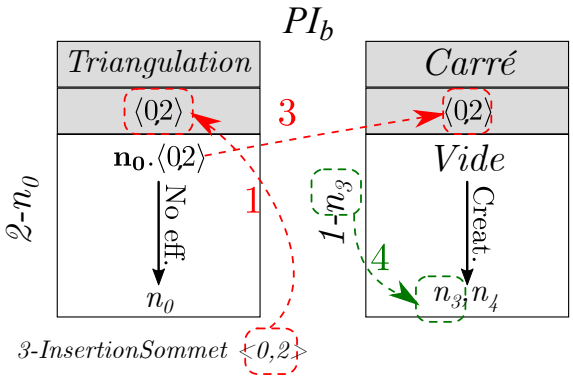


Figure B.11 : Journal d'historique modifié de PI_{2b}

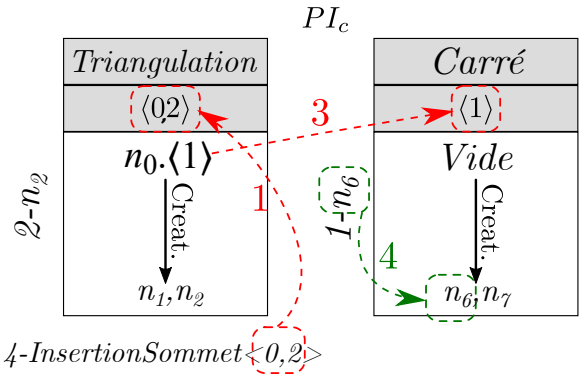


Figure B.12 : Journal d'historique modifié de PI_{3c}

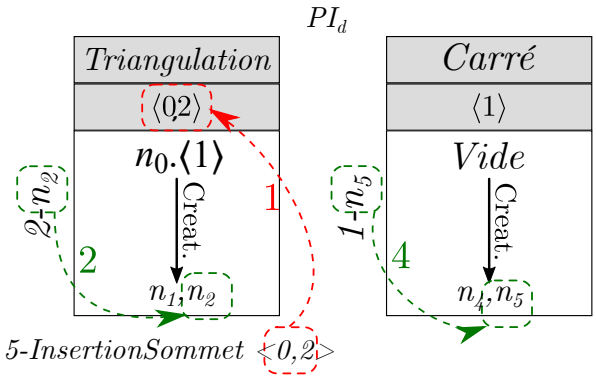


Figure B.13 : Journal d'historique modifié de PI_{4d}

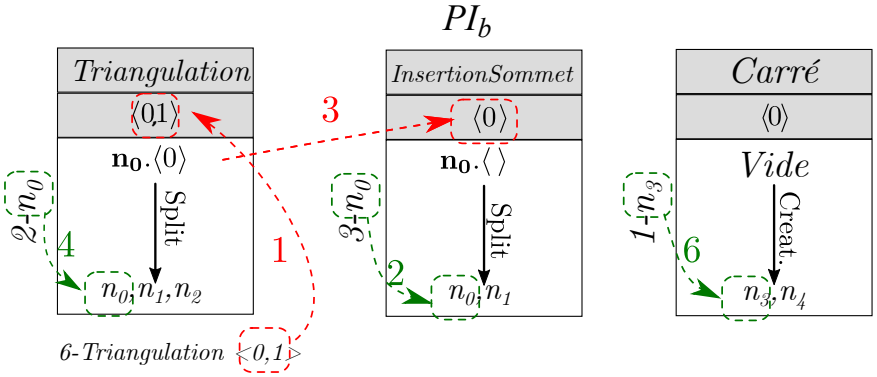


Figure B.14 : Journal d'historique modifié de PI_{5b}

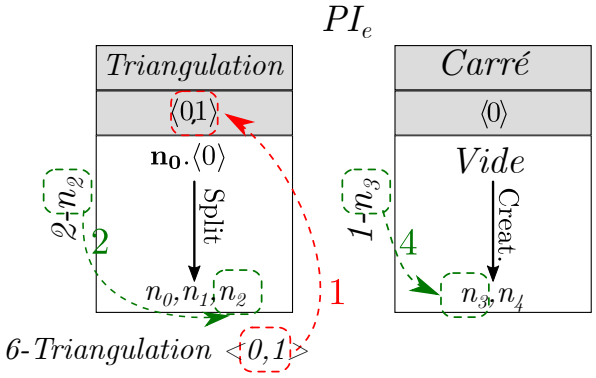


Figure B.15 : Journal d'historique modifié de PI_{5e}

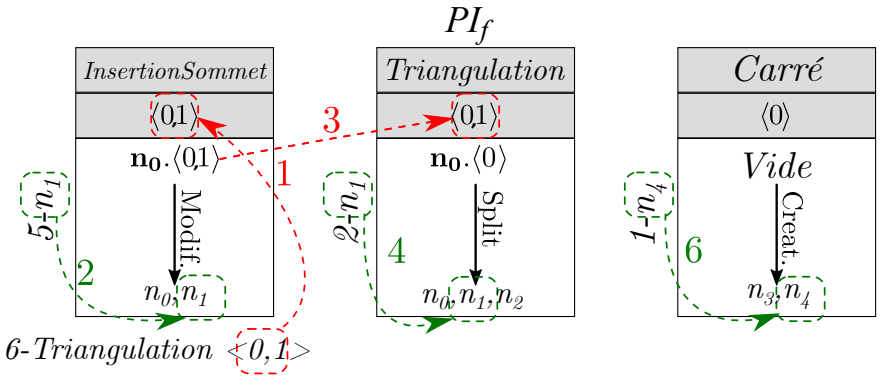


Figure B.16 : Journal d'historique modifié de PI_{5f}

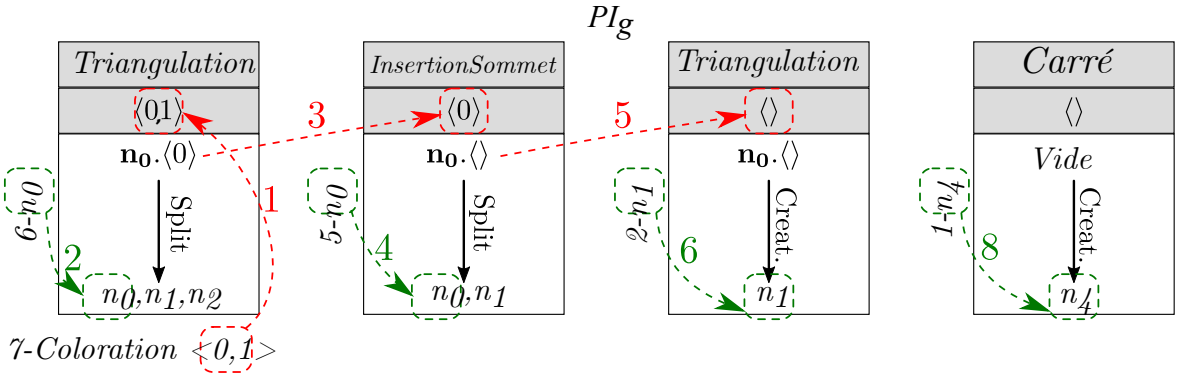


Figure B.17 : Journal d'historique modifié de PI_{6g}

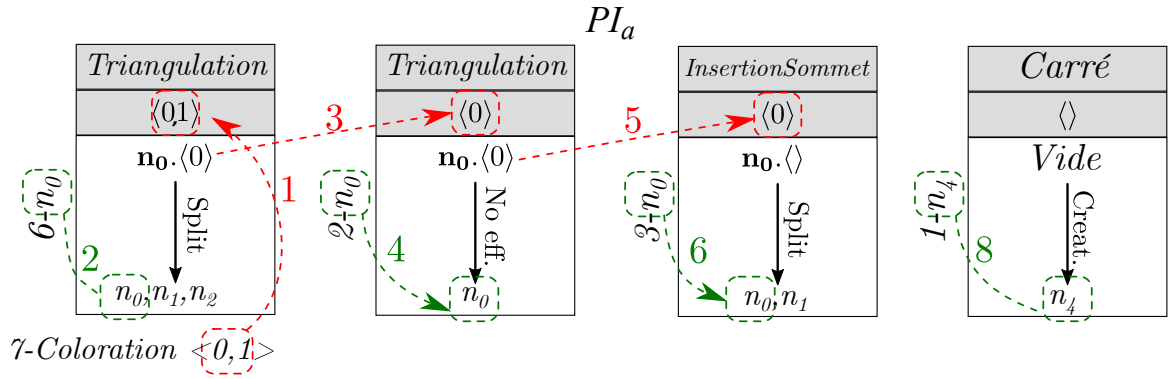


Figure B.18 : Journal d'historique modifié de PI_{7a}

Bibliographie

[AM95] R Anderl and R Mendgen. Parametric design and its impact on solid modeling applications. In *SMA '95 Proceedings of the third ACM symposium on Solid modeling and applications*, pages 1–12, Salt Lake City, USA, Mai 1995.

[AMP99] Dago Agbodan, David Marcheix, and Guy Pierra. A data model architecture for parametrics. *Journal for Geometry and Graphics*, 3(1) :17–38, 1999.

[BA10] Mehdi Baba Ali. *Système de nomination hiérarchique pour les systèmes paramétriques*. PhD thesis, Université de Poitiers, 2010.

[Bae69] R Baecker. Picture-driven animation. In *Proceeding AFIPS '69 (Spring) Proceedings of the May 14-16, 1969, spring joint computer conference*, pages 273–288, Boston, USA, Mai 1969.

[BALGB14] H. Belhaouari, A. Arnould, P. Le Gall, and T. Bellet. Jerboa : A graph transformation library for topology-based geometric modeling. In *International Conference on Graph Transformation*, page 269–284, York, UK, Juillet 2014.

[BAMS09] Mehdi Baba-Ali, David Marcheix, and Xavier Skapin. An edge matching technique for non-planar face intersections in geometric parametric models. In *IEEE International Conference on Shape*, pages 201–208, Beijing, Chine, Juin 2009.

[Bau75] Bruce Baugmart. A polyhedron representation for computer vision. In *AFIPS '75 Proceedings of national computer conference and exposition*, pages 589–596, Anaheim, USA, Mai 1975.

[Bel12] Thomas Bellet. *Transformation de graphes pour la modélisation géométrique à base topologique*. PhD thesis, Université de Poitiers, 2012.

[Ben79] Ian Benest. A review of computer graphics publications. *Computers and Graphics*, 4(2) :95–136, 1979.

[Bid05] Rafael Bidarra. A feature-based solution to the persistent naming problem. *Computer-Aided Design and Applications*, 2(1-4) :517–526, 2005.

[Ble] Fondation Blender. Blender. https ://www.blender.org/.

[Boe93] Wolfgang Boehm. An affine representation of de casteljau's and de boor's rational algorithms. *Computer Aided Geometric Design*, 10(3-4) :175–180, 1993.

[Bra75] Ian Braid. The synthesis of solids bounded by many faces. In *Communications of the ACM*, volume 18, New York, USA, Avril 1975.

[BS01] B. Bettig and J. Shah. Derivation of a standard set of geometric constraints for parametric modeling and data exchange. *Computer Aided Design*, 33(1) :17–33, 2001.

[BSBAM17] F. Ben Salah, H. Belhaouari, A. Arnould, and P. Meseure. A general physical-topological framework using rule-based language for physical simulation. In *12th International Conference on Computer Graphics Theory and Applications*, Porto, Portugal, Février 2017.

[But79] M. Buthion. Un programme qui résout formellement des problèmes de constructions géométriques. *RAIRO Informatique*, 13(1) :73–106, 1979.

[Bé86] Pierre Bézier. *Courbes et Surfaces*. HERMES, 1986.

[Car07] Olivier Carton. Langages formels, calculabilité et complexité, cours de l'ens. https://gaati.org/bisson/tea/lfcc.pdf, 2007.

[CCH96] Vallis Capoyleas, Xiangping Chen, and Christoph Hoffmann. Generic naming in generative, constraint- based design. *Computer-Aided Design*, 28(1) :17–26, 1996.

[CDFM+94] Paolo Cignoni, Leila De Floriani, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *Proceeding VVS '94 Proceedings of the 1994 symposium on Volume visualization*, pages 19–26, Tysons Corner, USA, Octobre 1994.

[CH94] Xhangping Chen and Christoph Hoffmann. Towards feature attachment. *Computer-Aided Design*, 27(9) :695–702, 1994.

[Che95] Xiangping Chen. *Representation, Evaluation and Editing of Feature-Based and Constraint-Based design*. PhD thesis, Purdue University, 1995.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3) :113–124, 1956.

[CMHK12] Sang-Uk Cheon, Duhwan Mun, Soonhung Han, and Byung Chul Kim. Name matching method using topology merging and splitting history for exchange of feature-based cad models. *Journal of Mechanical Science and Technology*, 26(10) :3201–3212, 2012.

[CMS18] Anaïs Cardot, David Marcheix, and Xavier Skapin. Persistent naming based on graph transformation rules to reevaluate parametric specification. In *Proceedings of CAD'18*, pages 387–391, Paris, France, Juillet 2018.

[Com] MAXON Computer. Cinema4d. https ://www.maxon.net/fr/produits/cinema-4d/cinema-4d/.

[Cor] Parametric Technology Corporation. Pro/engineer. https ://www.ptc.com/fr/products/cad/pro-engineer.

[dB78] Carl de Boor. *A Practical Guide to Spline*, volume 27. Springer-Verlag New York, 1978.

[DC85] P De Casteljau. *Formes et Poles*. HERMES, 1985.

[DMS98] Francois Dufourd, Pascal Mathis, and Pascal Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1) :73–119, 1998.

[dP CP] Université de Poitiers and École Centrale Paris. Jerboa. http://xlim-sic.labo.univ-poitiers.fr/jerboa/.

[EEP06] H. Ehrig, K. Ehrig, and U. Prange. *Fundamentals of Algebraic Graph Transformation*. Springer, 2006.

[EL93] Hervé Elter and Pascal Lienhardt. Different combinatorial models based on the map concept for the representation of subsets of cellular complexes. *Modeling in Computer Graphics*, pages 193–212, 1993.

[Fer86] David Ferguson. The construction of curves and surfaces using numerical optimization techniques. *Computer-Aided Design*, 18(1) :15–21, 1986.

[FHM16] Shahjadi Hisan Farjana, Soonhung Han, and Duhwan Mun. Implementation of persistent identification of topological entities based on macro-parametrics approach. *Journal of Computational Design and Engineering*, 3(2) :161–177, 2016.

[FMH15] Shahjadi Hisan Farjana, Duhwan Mun, and Soonhung Han. A method for persistent identification of topological entities in a parametric cad model. In *Proceedings of the 6th International Conference on manufacturing, Machine Design and Tribology*, Okinawa, Japon, Avril 2015.

[FMJ05] Sebti Foufou, Dominique Michelucci, and Jean-Paul Jurzak. Numerical decomposition of geometric constraints. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 143–151, Cambridge, USA, Juin 2005.

[FPDM14] Hetal Fitter, Akash Pandey, Patel Divyand, and Jitendra Mistry. A review on approaches for handling bezier curves in cad for manufacturing. *Procedia Engineering*, 97 :1155–1166, 2014.

[FPL16] Benjamin Fitch, Patrick Parslow, and Karsten Lundqvist. Evolving complete l-systems : Using genetic algorithms for the generation of realistic plants. In *Artificial Life and Intelligent Agents Symposium*, pages 16–23, Birmingham, UK, Juin 2016.

[GAB+16] V. Gauthier, A. Arnould, H. Belhaouari, S. Horna, M. Perrin, M. Poudret, and J.-F. Rainaud. A topological approach for automated unstructured meshing of complex reservoir. In *15th edition of the European Conference on the Mathematics of Oil Recovery*, Amsterdam, Pays Bas, Août 2016.

[Gal99] Jean Gallier. *Curves and Surfaces In Geometric Modeling : Theory And Algorithms*. Morgan Kaufmann, 1999.

[Gau18] Valentin Gauthier. *Reconstruction multi-échelle de géomodèles 3D à base de règles*. PhD thesis, Université de Poitiers, 2018. http ://theses.fr/s161702.

[GR74] William Gordon and Richard Riesenfeld. Bernstein- bezier methods for the computer-aided design of free-form curves and surfaces. In *Journal of the ACM*, volume 21, pages 293–310, New York, USA, Avril 1974.

[GSDL06] Carine Grasset-Simon, Guillaume Damiand, and Pascal Lienhardt. nd generalized map pyramids : definition, representations and basic operations. *Pattern Recognition*, 39(4) :527–538, 2006.

[HK01] C.M Hoffman and J Kimb. Towards valid parametric cad models. *Computer Aided Design*, 33(1) :81–90, 2001.

[HMGB09] S. Horna, D. Meneveaux, Damiand G., and Y. Bertrand. Consistency constraints and 3d building reconstruction. *Computer-Aided Design*, 41(1) :13–27, 2009.

[HMVG09] S. Haegler, P. Muller, and L. Van Gool. Procedural modeling for digital cultural heritage. *Journal on Image and Video Processing - Special issue on image and video processing for cultural heritage*, 2009.

[Inc] Side Effects Software Inc. Houdini. https ://www.sidefx.com/.

[Jac64] Edwin Jacks. A laboratory for the study of graphical man-machine communication. In *Proceedings of the October 27-29, 1964, fall joint computer conference, part I*, pages 343–350, San Francisco, USA, Octobre 1964.

[Kri95] J. Kripac. A mechanism for persistently naming topological entities in history based parametric solid models. In *Proceedings of the 3rd ACM symposium on Solid Modeling and Applications*, pages 21–30, Salt Lake City, USA, Mai 1995.

[KSSH66] J Koford, P. Strickland, G Sporzynski, and E Hubacber. Using a graphic data-processing system to design artwork for manufacturing hybrid integrated circuits. In *AFIPS '66 (Fall) Proceedings of the November 7-10, 1966, fall joint computer conference*, pages 229–246, San Francisco, USA, Novembre 1966.

[LFB07] Pascal Lienhardt, Laurent Fuchs, and Yves Bertrand. Modèles topologiques. *Informatique graphique, modélisation géométrique et animation*, pages 49–93, 2007.

[Lie94] Pascal Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Application*, 4(3) :275–324, 1994.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development i and ii. filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3) :280–299, 1968.

[LP16] Pascal Lienhardt and Samuel Peltier. Simploidal sets. Technical report, Université de Poitiers. XLIM-ASALI - Synthèse et analyse d'images, 2016.

[Man88] Martti Mantyla. *Introduction to Solid Modeling*. Computer Sience Press, 1988.

[MCSDG18] David Marcheix, Anaïs Cardot, Xavier Skapin, and Nadine Dieudonné-Glad. A persistent naming system based on graph transformation rules. *WSCG*, pages 48–54, Mai 2018.

[MF09] Dominique Michelucci and Sebti Foufou. Interrogating witnesses for geometric constraint solving. In *Proceedings of the 2009 ACM Symposium on Solid and Physical Modeling*, Francisco, USA, Octobre 2009.

[MH76] J Mazziotta and H Huang. Thread (three-dimensional reconstruction and display) with biomedical applications in neuron ultrastructure and computerized tomography. In *AFIPS '76 Proceedings of the June 7-10, 1976, national computer conference and exposition*, pages 241–250, New York, USA, Juin 1976.

[MH09] Duhwan Mun and Soonhung Han. Identification of topological entities and naming mapping for parametric cad model exchange. *International Journal of CAD/CAM*, 5(1) :69–81, 2009.

[MNS96] Martti Mäntylä, Dana S. Nau, and Jami J. Shah. Challenges in feature-based manufacturing research. *Communications of the ACM*, 39(2) :77–85, 1996.

[Moj] Mojang. Minecraft. https ://www.engadget.com/2015/03/04/how-minecraft-worlds-are-made/.

[Mou16] Adel Moussaoui. *Geometric Constraint Solver*. PhD thesis, Ecole nationale Supérieure d'Informatique (ex I.N.I), Alger, 2016.

[MP96] R. Mech and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410, New Orleans, USA, Aout 1996.

[MP02] David Marcheix and Guy Pierra. A survey of the persistent naming problem. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 13–22, Saarbrücken, Allemagne, Juin 2002.

[Owe91] J.C. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 397–407, Austin, USA, Juin 1991.

[Owe97] Jon Owen. *STEP : An Introduction*. Information Geometers Ltd, 1997.

[PHM93] Przemyslaw Prusinkiewicz, Mark Hammel, and Eric Mjolsness. Animation of plant development. In *Proceedings of SIGGRAPH 93*, pages 351–360, Anaheim, USA, Août 1993.

[Pou09] Mathieu 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, Université de Poitiers, 2009.

[PT97] L. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag Berlin Heidelberg, 1997.

[QB15] R. Quattrini and E. Baleani. Theoretical background and historical analysis for 3d reconstruction model : Villa thiene at cicogna. *Journal of Cultural Heritage*, 16(1) :119–125, 2015.

[Req77] Aristides Requicha. Mathematical models of rigid solid objects. Technical report, University of Rochester. Production Automation Project, 1977.

[Rol91] Dieter Roller. Advanced methods for parametric design. *Geometric Modeling*, pages 251–266, 1991.

[RS80] G. Rozenber and a. Saolmaa. *The Mathematical Theory of L System*. Springer, Boston, MA, 1980.

[Sch93] P. Schreck. *Automatisation des constructions géométriques à la règle et au compas*. PhD thesis, Université de Strasbourg I, 1993.

[SM95] Jami J. Shah and Martti Mäntylä. *Parametric and feature-based CAD/CAM : concepts, techniques, and applications*. John Wiley Sons, 1995.

[Smi84] Alvy Ray Smith. Plants, fractals, and formal languages. *ACM SIGGRAPH Computer Graphics*, 18(3) :1–10, 1984.

[Sofa] Bethesda Softworks. Arena. https ://procedural-generation.tumblr.com/post/129107920408/the-elder-scrolls-arena-1994-which-game-lets.

[Sofb] Bethesda Softworks. Oblivion. https://archive.rpgamer.com/games/elderscrolls/elder4/elder4interview.html.

[Sut63] Ivan Sutherland. Sketchpad : A man-machine graphical communication system. In *Proceeding DAC '64 Proceedings of the SHARE design automation workshop*, volume 6, pages 329–346, Detroit, USA, 1963.

[Sysa] Dassault Systèmes. Abaqus. https ://www.3ds.com/fr/produits-et-services/simulia/produits/abaqus/.

[Sysb] Dassault Systèmes. Catia. https ://www.3ds.com/fr/produits-et-services/catia/.

[TGMD09] O. Terraz, G. Guimberteau, D. Merillou, S.and Plemenos, and Ghazanfar D. 3gmap l-systems : an application to the modelling of wood. *The Visual Computer*, 25(2) :165–180, 2009.

[The69] C Theiss. Computer graphics displays of simulated automobile dynamics. In *AFIPS '69 (Spring) Proceedings of the May 14-16, 1969, spring joint computer conference*, pages 289–296, Boston, USA, Mai 1969.

[TW13] Sean Tessier and Yang Wang. Ontology-based feature mapping and verification between cad systems. *Advanced Engineering Informatics*, 27(1) :76–92, 2013.

[Vin83] A. Vince. Combinatorial maps. *Journal of Combinatorial Theory Series B*, 34 :1–21, 1983.

[WZZ01] Junjun Wu, Tianbing Zhang Zhang, and Ji Zhou. A face based mechanism for naming, recording and retrieving topological entities. *Computer-Aided Design*, 33(10) :687–698, 2001.

[YMJMSMC14] E. Yeguas, M.J. Marín-Jiménez, R. Muñoz-Salinas, and R. Medina-Carnicer. Conflict-based pruning of a solution space within a constructive geometric constraint solver. *Applied Intelligence*, 41(3) :897–922, 2014.

Index

Table des figures

Directeur de thèse : Pascal Lienhardt Anaïs Cardot

Rejeu basé sur des règles de transformation de graphes

Abstract

Réaliser des variations d'un même modèle est un besoin en expansion dans de nombreux domaines de modélisation (architecture, archéologie, CAO, etc.). Mais la production manuelle de ces variations est fastidieuse, il faut donc faire appel à des techniques permettant de rejouer automatiquement tout ou partie du processus de construction du modèle, après spécification des modifications.

La majorité des approches dédiées à la réalisation du rejeu sont basées sur un système de modélisation paramétrique, composée d'un modèle géométrique et d'une spécification paramétrique permettant d'enregistrer la succession d'opérations l'ayant créé ainsi que leurs paramètres. On peut ensuite faire varier ces paramètres ou éditer cette liste d'opérations afin de modifier le modèle. On utilise pour cela un système de nommage persistant, introduit dans les années 90, et permettant d'identifier et d'apparier les entités d'une spécification initiale et celles d'une spécification rejouée.

L'objectif de cette thèse est de proposer un système de nommage persistant général, homogène et permettant de gérer l'édition de spécification paramétriques (déplacer, ajouter et supprimer des opérations). Nous nous basons sur la bibliothèque Jerboa, qui repose sur des règles de transformation de graphes, tant pour utiliser ces règles dans la réalisation de la méthode de nommage que pour lier les notions de spécification paramétrique à ces règles de transformations de graphes.

Nous décrivons ensuite comment exploiter notre méthode de nommage pour rejouer et éditer des spécifications paramétriques d'opérations, puis nous la comparons avec les approches de la littérature.

Mots clés : Spécification paramétrique, Rejeu, Nommage persistant, Règles de transformation de graphe, Cartes généralisées

In many modelling fields, such as architecture, archaeology or CAD, performing many variations of the same model is an expanding need. But building all those variations manually takes time. It is therefore needed to use automatic technics to revaluate some parts of a model, or even an entire model, after the user specifies the modifications.

Most of the existing approaches dedicated to revaluating models are based on a system called parametric modelling. It is made of two parts, a geometric model and a parametric specification, which allows to record the series of operation that created the model, and the different parameters of those operations. This way, the user can change some parameters, or edit the list of operations to modify the model. To do so, we use a system called persistent naming, introduced during the 90ies, that allows us to identify and match the entities of an initial specification and the ones of a revaluated specification.

In this thesis, our goal is to propose a persistent naming system that would be general, homogeneous and that would allow the user to edit a parametric specification (which means move, add, or delete some operations). We base our system on the Jerboa library, which uses graph transformation rules. This way, we will be able to use those rules to create our naming system, while also linking the notions of graph transformation rules and parametric specification.

We will then describe how to use our naming method to revaluate or edit parametric specifications. Finally, we will compare our method with the other ones from the literature.

Keywords : Parametric specification, Reevaluation, Persistent naming, Graph transformation rules, Generalized map