L'approche catégorique des grammaires de graphes

Gradit Pierre

LAAS/CNRS,
7 Avenue du Colonel Roche,
31077 TOULOUSE CEDEX 7

Résumé Une difficulté de la spécification de systèmes dynamiques est qu'un état atteint par le système dépend de la politique d'allocation de références. L'approche algébrique des grammaires de graphes pose explicitement le problème et caractérise les états identiques. L'utilisation de la notion de morphisme pour décrire, non seulement l'instanciation, mais la règle elle-même[3] permet cette caractérisation.
L'autre interêt de cette approche est d'étendre la notion de graphe à toute algèbre finie sur une signature ne contenant que des opérateurs unaires[9]. Cette définition donnée permet, non seulement de relier les noeuds par des arêtes, mais de les doter d'attributs. En outre, une telle signature peut s'interpréter et être manipulée comme un graphe dirigé.

Introduction

Dans les systèmes dynamiques, où le nombre de composants évolue au cours du déroulement, le nommage des composants est assujetti à une politique d'allocation de références. Deux spécifications identiques mais instrumentées par deux politiques différentes d'allocation vont donner deux exécutions différentes que l'on peut corréler. La notion d'isomorphisme permet de rendre compte de cette corrélation, par l'idée qu'il véhicule de permutation sur les références préservant la structure. L'approche algébrique des grammaires de graphes constitue un outil puissant pour spécifier ces systèmes car la notion de morphisme est au coeur de cette théorie. Elle utilise l'abstraction et les notations de la théorie des catégories où un morphisme entre deux objets est dénoté par une flèche reliant ces objets. Dans cette approche, l'état du système est un objet et une transformation de ce système correspond à une flèche $\texttt{transformation} : \texttt{avant} \rightarrow \texttt{apres}$. L'intérêt réside dans la possibilité de *dériver* d'une flèche $\texttt{regle} : \texttt{avant} \rightarrow \texttt{apres}$ un ensemble potentiellement infini de transformations[5], par l'utilisation d'une construction appelée *pushout*. Deux approches existent au sein de l'approche catégorique des grammaires de graphes. L'approche qui considère une règle comme deux morphismes totaux ayant même source qui va nécessiter deux *pushouts* pour sa dérivation[6,5], et l'approche qui considère une règle comme un morphisme partiel ne nécessitant qu'un *pushout* pour sa dérivation[9,3]. Nous expliciterons un contexte où ces deux approches coïncident tout en travaillant dans l'approche par simple-pushout qui permet de construire effectivement toutes ces flèches comme des morphismes entre algèbres finie.

Le travail séminal de [9] contient 1. la définition des morphismes partiels de graphes et du pushout dans ce contexte 2. la caractérisation des signatures pour lesquelles le pushout de morphismes partiels (i.e. la dérivation) existe toujours. Le premier point a été largement intégré dans la présentation catégorique des grammaires de graphes (voir la présentation commune dans [5]). Par contre, la caractérisation est restée un point secondaire, alors qu'elle contient de notre point de vue les germes d'une théorie prometteuse. Cette caractérisation est la suivante: la dérivation de morphisme partiel est complète pour toute algèbre d'une signature ne contenant que des opérateurs unaires [9]. Cette caractérisation est un progrès considérable car elle permet de définir la structure adaptée sans avoir à recourir à la surcharge de la structure fixée par le formalisme (e.g. représenter l'étiquetage des noeuds par des boucles [5]). On s'affranchit par là même d'une structure de graphes par défaut et donc de son choix toujours problématique.

Notre contribution à cette approche est de nature réflective. Son point de départ consiste à remarquer qu'une signature de graphe est syntaxiquement un "support de diagramme". Cette remarque prend tout son sens en considérant ce diagramme dans la catégorie **Set** (c'est à dire associer à chaque noeud un ensemble, à chaque flèche une fonction totale). En effet c'est syntaxiquement une algèbre de la signature correspondant au "support de diagramme". Ceci permet d'une part une gestion des signatures avec les mêmes outils que ceux utilisé pour définir les graphes dirigés et d'autre part de se poser la question: que recouvre une transformation de "support de diagramme"?

Notre travail s'articule comme suit: la première partie fournit les bases catégoriques nécessaires. La deuxième définit les graphes et leurs transformations en suivant l'approche de [9]. Notre contribution est développée dans la partie trois où l'on exploite le parallèle entre signature et "support de diagramme". Nous concluons en décrivant les travaux en cours et les perspectives de ce travail.

1 Petit précis catégorique

La théorie des catégories débute par l'observation qu'un grand nombre de propriétés des systèmes mathématiques peuvent être unifiées et simplifiées par une *présentation* à base de diagrammes de flèches. La définition d'une catégorie est donc à la base un ensemble d'objets et de flèches identifiés sans ambiguité et doté d'opérateurs spécifiques, la flèche d'identité et la composition de flèches.

Definition 1 (Catégorie). *Une catégorie* **C** *est*

1. *Une collection d'objets, dénotés par a,b,c...*
2. *Une collection de flèches, dénotés par f,g,...*
3. *Deux opérateurs reliant les flèches aux objets:* $\mathbf{dom}(\_)$ *qui assigne à chaque flèche son objet source et* $\mathbf{cod}(\_)$ *qui assigne à chaque flèche son objet de destination*
4. *Un opérateur de composition qui assigne à chaque paire de flèche* $\langle g : b \to c, f : a \to b \rangle$ *telle que* $\mathbf{dom}(g) = \mathbf{cod}(f)$ *une flèche* $g \circ f$ *appelée leur composition, avec* $g \circ f : \mathbf{dom}(f) \to \mathbf{cod}(g)$. *Cet opérateur étant associatif:*

   Soit des objets et des flèches dans la configuration suivante: $a \xrightarrow{f} b \xrightarrow{g} c \xrightarrow{k} d$, *on a toujours l'égalité* $k \circ (g \circ f) = (k \circ g) \circ f$.
5. *Un opérateur* $\mathtt{Id}^-$ *qui assigne à chaque objet a une flèche* $1_a = \mathtt{Id}^a : a \to a$; *vérifiant la loi de l'unité:*

   Toute composition des flèches $f : a \to b$ *et* $g : b \to c$ *avec la flèche identité* $\mathtt{Id}^b$ *donne:* $\mathtt{Id}^b \circ f = f$ *et* $g \circ \mathtt{Id}^b = b$.

Dans la suite on travaillera principalement avec quatre catégories qui ont pour objets les ensembles finis. Elles sont données ici par ordre croissant pour l'inclusion sur l'ensemble des flèches.

Relation qui a comme flèches les relations (i.e. partie du produit cartésien du domaine et du codomaine)

Co-injection qui a comme flèches les fonctions partielles (appellée co-injections car la relation inverse d'une fonction partielle est injective).

Fonctions qui a comme flèches les fonctions totales (i.e. **Set**[10]).

Bijections qui a comme flèches les bijections.

1.1 Diagrammes

On représente rarement une catégorie dans son intégralité, mais on exprime des propriétés que l'on pourrait qualifier de locales par l'usage de diagrammes.

Definition 2 (Diagrammes). *Un diagramme dans une catégorie* **C** *est un graphe dirigé étiqueté par des objets et des flèches de* **C** *de façon consistante.*

La consistance signifie que si une flèche du graphe est étiquetée par une flèche de la catégorie ayant comme domaine A et comme co-domaine B alors les extrémités de cette flèches doivent être A et B.

Les diagrammes sont une façon compacte d'exprimer des propriétés sur des catégories. On considère la convention suivante, appelée "commutation", par défaut dans les diagrammes que l'on considère.

Definition 3 (Commutation). *Un diagramme dans une catégorie* **C** *commute si pour toute paire de sommets du diagramme, tous les chemins (vu comme des flèches par composition) menant de A à B sont égaux à condition qu'au moins un soit de longueur supérieure à 1.*

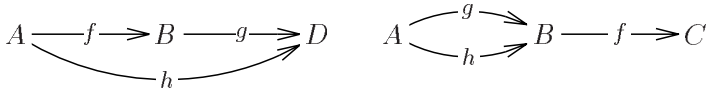


FIG. 1 –. *Exemples de diagrammes commutants.*

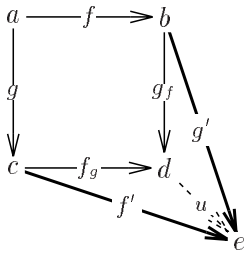
Considérons les deux diagrammes donnés figure 1. Que le diagramme de gauche commute signifie que pour les sommets A et D tous les chemins menant de A à B sont égaux: i.e. $g \circ f = h$. Par contre que le diagramme de droite commute signifie que $f \circ h = f \circ g$ mais ne donne aucune information sur g et h. Dans la catégorie **Fonctions**, g et h sont égaux si et seulement si f est surjective.

1.2 Constructions universelles

Un fondement de la théorie des catégorie est l'identification d'éléments universaux dans des collections d'objets ou de flèches. Pour illustrer cette notion d'universalité, on prend la notion d'élément initial (resp. terminal). Un élément est initial (resp. terminal) dans une collection si et seulement si il existe une unique flèche venant (resp. menant) de cet élément vers chaque élément de la collection. Par exemple dans **Fonctions**, l'ensemble vide ($\emptyset$) est initial est un ensemble à un élement (noté $*$) est terminal.

Parmi les constructions universelles, une seule nous sera réelement utile: le pushout. En effet c'est un pushout qui traduira le concept de dérivation (Cf Définition 6) des règles en transformation.

Définition 1 (Pushout (co-pullback)) *Le pushout*
d'une paire de flèches $\langle f : a \to b, g : a \to b \rangle$
est une paire de flèches $\langle g_f : b \to d, f_g : c \to d \rangle$
vérifiant $g_f \circ f = f_g \circ g$
telle que pour toute paire de flèches $\langle g' : b \to e, f' : c \to e \rangle$
vérifiant $g' \circ f = f' \circ g$
il existe une unique flèche $u : d \to e$
telle que $u \circ g_f = g'$ et $u \circ f_g = f$.



Une interprétation graphique est donnée par la figure à droite de la définition. La notation graphique utilisée s'inspire de [2], les flèches épaisses dénotent le quantifieur universel et les pointillés dénotent le quantifieur existentiel (et l'unicité). Il est important de noter qu'un pushout existe independemment de la façon dont on le construit, La figure 2 donne différent diagrammes "carrés" étiquetés dans **Fonctions** (i.e **Set**) qui commutent. Tous ces "carrés" sont obtenu en complétant les deux flèches un et m. La caractérisation du pushout permet de dire que C_2 est un pushout car C_2 est le seul qui vérifie la propriété. En effet pour C_1 la partie universelle n'existe pas (vers C2 par exemple), et pour C_3 elle n'est pas unique, et ce vers lui même. En effet, c peut être affecté indifféremment à a, b ou c, car n'étant pas "concerné" par la commutation.

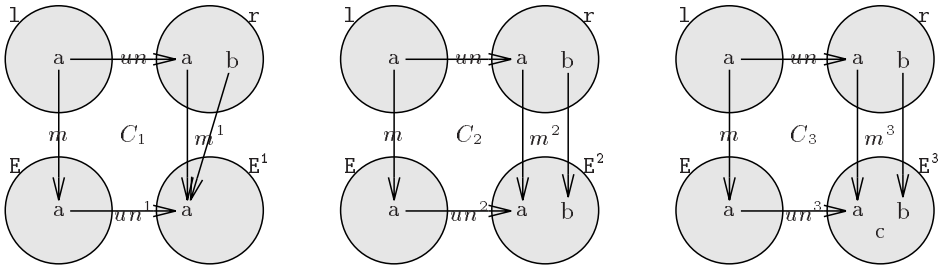


FIG. **2** –. *Différents "carrés" commutant possible à partir de D*

2 Approche Algébrique des graphes et de leur transformation

Cette partie s'attache à définir les graphes de façon algébrique et donne les définitions de transformations de graphes, de collage de graphe ainsi que d'équivalence de graphes. Le point de départ de ces définitions est la définition élémentaire des algèbres universelles : la signature multi-sorte.

Définition 2 (Signature multi-sortes[7]) *Une signature* $\mathcal{S} \stackrel{def}{=} (\Sigma, \Omega, \eta)$ *est la donnée de* S*, un ensemble de sortes;* $\mathtt{op}_S$*, un ensemble de Symboles fonctionnels;* $\eta : \Omega \to \Sigma^* \times \Sigma$*, une fonction qui à un opérateur associe le couple (sortes des opérandes, sorte du résultat).*

Introduisons à présent la clé de voûte de l'édifice:

Définition 3 *Une signature de graphe est une signature multi-sorte ne contenant que des opérateurs unaires [9].*

Cette définition permet de simplifier la définition de la fonction $\eta : \Omega \to \Sigma \times \Sigma$, et donc de "casser" cette fonction en deux fonctions $\alpha : \Omega \to \Sigma$ et $\omega : \Omega \to \Sigma$. De ce qui précède on déduit:

Corollaire 1 (Signature de graphe) *Une signature de graphe est une paire de paires* $\mathcal{G} \stackrel{def}{=} ((\Sigma, \Omega), (\alpha, \omega))$ *où*

- Σ *est un ensemble dont les éléments sont appelés sortes.*
- Ω *est un ensemble dont les éléments sont appelés opérateurs*
- α *est une fonction (totale) de* Ω *dans* Σ *appelée domaine.*
- ω *est une fonction (totale) de* Ω *dans* Σ *appelée co-domaine.*

La première paire est une paire d'ensembles et la deuxième paire une paire de fonctions du deuxième ensemble vers le premier.

Définition 4 ($\mathcal{G}$-Algèbres[7]) *Un élément de* $\texttt{algebres}(\mathcal{G})$ *(une $\mathcal{G}$-algèbre) est une structure* $A_{(\mathcal{G})} \stackrel{def}{=} (A, \mathbf{A})$ *avec*

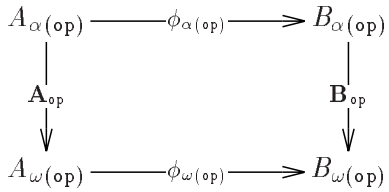
- $A \stackrel{def}{=} (A_{\mathtt{s}})_{\mathtt{s}\in\Sigma}$ *une famille d'ensembles (i.e. objets de* **Fonctions***) Σ-indexée. Nous appellerons chaque $A_{\mathtt{s}}$ ensemble porteur de la sorte* $\mathtt{s}$.
- $\mathbf{A} \stackrel{def}{=} (\mathbf{A}_{\mathtt{op}})_{\mathtt{op}\in\Omega}$, *une famille de fonctions "totales"(i.e. flèches de* **Fonctions***) Ω-indexée telle que* $\mathbf{A}_{\mathtt{op}} : A_{\alpha(\mathtt{op})} \to A_{\omega(\mathtt{op})}$.

Une sous-algèbre $B_{(\mathcal{G})} \subseteq A_{(\mathcal{G})}$ est une $\mathcal{G}$-algèbre vérifiant $\forall \mathtt{s} \in \Sigma, B_{\mathtt{s}} \subseteq A_{\mathtt{s}}$.

2.1 Morphismes paramétrés de graphes

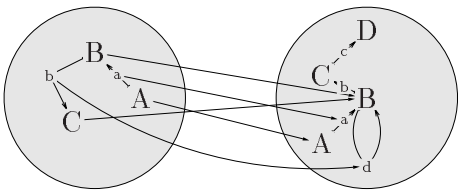
Un morphisme de graphe est une famille de "relations" entre les ensembles porteurs du graphe source vers les ensembles porteurs du graphe destination préservant la structure de graphe. Différents morphismes sont utilisés dans notre approche, pour factoriser les définitions on va paramétrer la définition par la sous-catégorie de **Relation** par laquelle on va "projeter" les ensembles porteurs. Une fois la définition factorisée on va l'interpréter pour les trois valeurs possibles du paramètre. Les différentes classes vont être les collages, les équivalences et les transformations.

Definition 4 ($\mathcal{G}$-Morphismes de classe `parametre`).
Soit $\texttt{parametre} \in \{\textbf{Bijection}, \textbf{Fonction}, \textbf{Co-injection}\}$.
Soit A et B deux $\mathcal{G}$-algèbres.
Un $\mathcal{G}$-morphisme de classe `parametre` $\phi : A \to B$ *est une famille* $(\phi_{\mathtt{s}} : A_{\mathtt{s}} \to B_{\mathtt{s}})_{\mathtt{s}\in\Sigma}$ *Σ-indexée de flèches de* `parametre` *telle que, pour tout opérateur* $\mathtt{op}$*de $\mathcal{G}$, le diagramme ci-dessous commute dans* **Relation** *(i.e.* $\phi_{\omega(\mathtt{op})} \circ \mathbf{A}_{\mathtt{op}} = \mathbf{B}_{\mathtt{op}} \circ \phi_{\alpha(\mathtt{op})}$*)*

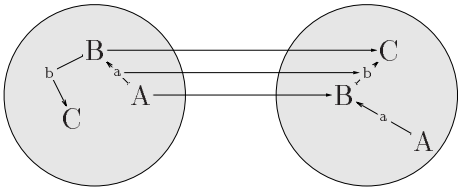


Cette construction se décine des trois façons suivantes:

$\mathcal{G}$-equivalence un $\mathcal{G}$-morphisme de classe **Bijection**.
$\mathcal{G}$-collage un $\mathcal{G}$-morphisme de classe **Fonction** [3,5]. L'exemple suivant est un Graphes-collage.



$\mathcal{G}$-transformation (ou $\mathcal{G}$-règle) un $\mathcal{G}$-morphisme de classe **Co-injection**[9]. La figure ci-dessous donne un exemple de Graphes-règle. Compte-tenu des équivalences précédemment exposés, on peut considérer des $\mathcal{G}$-graphes ou des $\mathcal{G}$-algèbres. L'exemple suivant est de signature Graphes.



Definition 5 (Catégorie parametre[$\mathcal{G}$]). *On définit la catégorie* parametre*[$\mathcal{G}$] ayant comme objet les $\mathcal{G}$-graphes et comme flèches les $\mathcal{G}$-morphismes de classe* parametre*.*

Ainsi l'ensemble des notions dont nous avons besoin, collage, transformation et équivalence sont différentes classes de morphismes entre algèbres de signature unaires.

2.2 Dérivation de transformation

L'intérèt d'un grammaire à base de règles et de disposer d'un mécanisme permettant de *dériver* une règle en un ensemble potentiellement infini de transformations[5] paramétrées par les collages utilisés pour les instancier. La présentation suit celle traditionnellement utilisée pour les réseaux de Petri: les conditions d'application (i.e. de sensibilisation) suivi de l'effet d'une règle applicable (i.e. du tir).

Définition 5 (Conditions d'application d'une règle) *Une $\mathcal{G}$-règle $p : L \to R$ est X-applicable sur un $\mathcal{G}$-graphe G ssi Un $\mathcal{G}$-collage $m : L \to G$ est une X-application de la règle*

i)njective *ssi pour tout opérateur* op $\in$ operateur*,*
$\forall x, y \in L_{\alpha(\mathtt{op})}, m_{\alpha(\mathtt{op})}(x) = m_{\alpha(\mathtt{op})}(y) \implies x = y$ *ou* $x, y \in (R_r)_{\alpha(\mathtt{op})}$*. Si le collage de deux éléments distincts coïncident, ils sont tous deux conservés par l'application de la règle.*

c)omplète *ssi pour tout opérateur* op $\in$ operateur*, pour tout élément* $o \in G_{\alpha(\mathtt{op})}$*,* $o(\mathtt{op}) \in m_{\omega(\mathtt{op})}((L - R_r)_{\omega(\mathtt{op})}) \implies o \in (L - R_r)_{\alpha(\mathtt{op})}$*. Si un élément est détruit par la règle, tout élément qui pointe sur lui via un opérateur est aussi détruit explicitement par la règle.*

Un **ci***-collage est un collage* **i***njectif et* **c***omplet.*

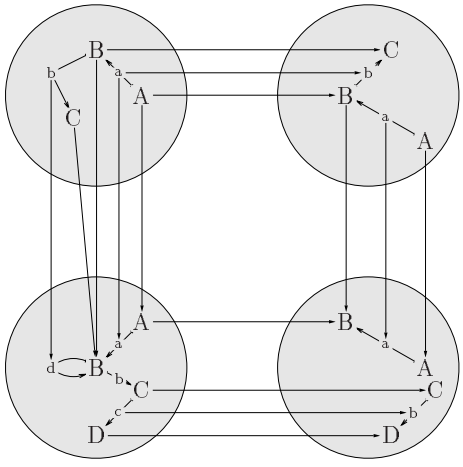


FIG. **3** –. *Dérivation d'une règle par un collage ni (c) ni (i)*

La dérivation correspond à la complétion du diagramme obtenu de façon universelle par un pushout dans la catégorie **Co-injection**$[\mathcal{G}]$.

Définition 6 (Effet d'une règle) *Une $\mathcal{G}$-transformation $p_m : G \to H$ est la X-derivation de $p : L \to R$ selon $m : L \to G$ ssi $m : L \to G$ définit une X-application de $p : L \to R$ et $(\langle p_m : G \to H, m_p : R \to H\rangle, univ)$ est un pushout de $\langle m : L \to G, p : L \to R\rangle$.*

La dérivation par simple-pushout est complète(i.e. toujours possible pour tout collage), en l'absence de la **ci***-condition le pushout détruit tous les éléments pointant sur un élément détruit comme dans l'exemple donné figure 3.*

Dans tous les cas, le pushout à la propriété suivante: p_m et f_g jointes sont surjectives, c'est à dire $p_m(G) \cup m_p(R) = H$. Ceci nous donne la propriété suivante,

Propriété 1 (Unicité de la dérivation modulo isomorphisme) *Si deux dérivations sont possibles $p_m : G \to H$ et $p'_m : G \to H'$, il existe une $\mathcal{G}$-equivalence $u : H \to H'$.*

L'universalité des deux pushout donne deux $\mathcal{G}$-transformations:$u = univ(\langle m : L \to G, p'_m : G \to H'\rangle) : H' \to H$, et $u' = univ'(\langle m : L \to G, p_m : G \to H\rangle) : H \to H'$. La surjectivité jointe des deux paires nous donne que u et u' sont surjectives pour avoir l'égalité des compositions. Donc u est l'équivalence recherchée.

2.3 Allocation de références: Pushout dans Fonctions

Le calcul du pushout peut se décomposer en deux parties distinctes[11]: un co-produit suivi d'un co-egaliseur. La formule permettant de combiner les deux est donné dans [11] pour le cas plus général des co-limites.

Définition 7 (Co-Produit) *Le co-produit d'une paire d'objets $\langle a, b\rangle$ est une paire de flèches $\langle j_a : a \to a + b, j_b : b \to a + b\rangle$ telle que pour toute paire de flèches $\langle f : a \to c, g : b \to c\rangle$ il existe une unique flèche $u : a + b \to c$ telle que $u \circ j_a = f$ et $u \circ j_b = g$*

Définition 8 (Co-Egaliseur) *Le co-égaliseur d'une paire de flèches $\langle f : a \to b, g : a \to b\rangle$ est une flèche $[f, g] : b \to c$ vérifiant $[f, g] \circ f = [f, g] \circ g$ telle que pour toute flèche $h : b \to d$ vérifiant $h \circ f = h \circ g$ il existe une unique flèche $u : d \to c$ telle que $u \circ [f, g] = h$.*

Dans **Fonctions** Le co-egaliseur est donné par le quotient de b par la plus petite relation d'équivalence contenant les paires $\langle fa, ga\rangle$[10,11], et on construit la flèche de b vers c en associant un élément à sa classe. Le choix de l'élément représentatif de la classe peut-être canonique (e.g. minimal) dès que l'ensemble de référence est *de facto* ordonné. Ce n'est plus le cas pour le co-Produit, et ceci est plus interessant dans notre problématique. Pour le co-produit on peut considérer différentes politiques permettant de le construire. On peut caractériser les flèches du co-produit comme étant des injections qui jointes doivent être surjectives. On construit donc le co-produit à l'envers en donnant la formule des inclusions et en déduisant l'ensemble destination. On suppose que l'ensemble des référence sous-jacentes est $I\!N$.

Par décalage Une des flèches du coProduit est la multiplication par deux, l'autre la multiplication par deux plus un (pair/impair). Ainsi chaque ensemble de départ s'injecte dans le nouveau. Outre la multiplication par deux de la plus grande référence, une propriété intéressante est absente, un noeud inchangé change de référence.

Par maximum Une des flèches est privilégiée (la future flèche "dérivée") pour laquelle l'injection est une "pseudo-identité" (une identité pas forcement surjective), et l'autre est obtenue en ajoutant aux références non-privilégiés le maximum des références privilégiés. Cette stratégie est la stratégie la plus couramment employée dans toutes les approches où les ensembles

de définitions sont dynamiques. Cette politique garantit qu'un noeud inchangé garde son identité.

Ramasse-miettes On peut raffiner la précédente en considérant que l'on permet aux références non-privilégiées (i.e. nouvelles) de s'injecter dans les trous de la "pseudo-identité". Cela rend assez finement compte d'un ramasse-miettes par la réutilisation de références qui furent utilisées mais qui ne le sont plus.

L'accumulation de ces stratégies montre bien qu'une implémentation particulière va donner des dérivations particulières. Mais le résultat de cette dérivation est toujours à un isomorphisme près équivalent à celle que l'on aurait obtenue en prenant une autre implémentation et ceci par l'utilisation de la propriété 1. Cette propriété peut s'étendre à des séquences de dérivations[5].

2.4 Pushout dans Fonctions[$\mathcal{G}$]

La définition du pushout dans **Fonctions**[$\mathcal{G}$] se fait en deux étapes, d'abord on réalise le pushout séparement sur chaque ensemble porteur. Ensuite, on déduit de façon unique les opérations à partir des pushouts sur les ensembles porteurs par l'utilisation de l'universalité des pushouts source.

Pour toute opérateur $\mathtt{op} : \mathtt{i} \to \mathtt{j}$ (i.e $\alpha(\mathtt{op}) = \mathrm{i}$, $\omega(\mathtt{op}) = \mathrm{j}$), cette construction est représentée par le diagramme donné figure 4 où : 1.La commutation des trapèzes supérieur et gauche sont des conséquences des $\mathcal{G}$-morphismes impliqués 2. les carrés intérieur et extérieur sont des pushouts par construction, d'où on déduit leur commutation et la propriété universelle. La complétion par la propriété du pushout extérieur(les sources) donne la construction souhaitée, La commutation s'obtenant en passant par le pushout intérieur (les destinations).

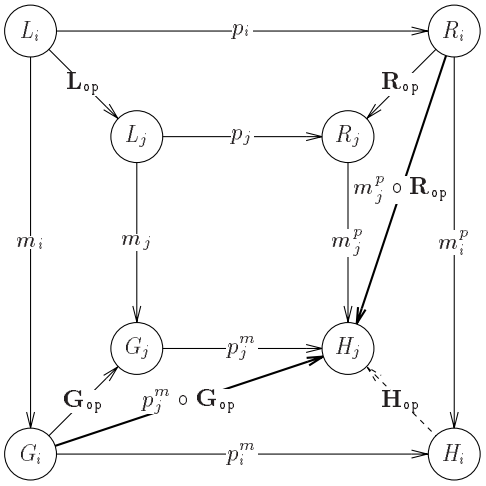


FIG. 4 –. *Construction du pushout dans la catégorie* **Fonctions**[$\mathcal{G}$].

2.5 Pushout dans Co-injections[$\mathcal{G}$]

Pour réaliser le pushout dans cette catégorie, deux possibilités soit on prend la construction générale de [9,3] qui repose sur un pushout dans **Fonctions**[$\mathcal{G}$], soit on travaille dans $(* \to$ **Fonctions**$)[\mathcal{G}]$, c'est les fonctions sur des ensembles où un élément est priviligié (généralement noté $\perp$). Ceci est exactement un cas d'application d'une construction de la théorie des catégorie, les catégories comma.

Definition 6 (Catégories comma). *Soit* **C** *une catégorie et c un objet de cette catégorie, on forme la catégorie* $(\mathbf{C} \to c)$ *(resp.* $(c \to \mathbf{C})$*) en prenant comme objet les paires* $\langle b, f : b \to c \rangle$

(resp. $\langle f : c \to b, b\rangle$), et comme flèches $h : \langle b, f\rangle \to \langle b', f'\rangle$ les flèches de **C** *vérifiant $h \circ f' = f$ (resp. $h : \langle f, b\rangle \to \langle f', b'\rangle$ les flèches de* **C** *vérifiant $h \circ f = f'$).*

Soit $* = \{\perp\}$ un ensemble à un élément, la catégorie (**Fonctions** $\to *$) est équivalente à la catégorie **Fonctions** car * est élément terminal de **Fonctions** (et donc la contrainte d'avoir un morphisme partant d'un ensemble vers $\perp$ n'en est pas une). Par contre ($* \to$ **Fonctions**) est équivalente à **Co-injections** si on représente les fonctions partielles comme des fonctions (totales) dans des ensembles dont un élément est pointé (image de $\perp$ par f) et où toutes les fonctions $g : \langle f : * \to c, c\rangle \to \langle f' : * \to c', c'\rangle$ vérifient $g(f(\perp)) = f'(\perp)$.

Cette équivalence s'étend pour les algèbres d'une signature où aucune suite infini d'opérateurs n'existe dont la destination du $i^{\text{ème}}$ coïncide avec la source du $(i+1)^{\text{ème}}$. Une telle signature est appelée hiérarchique.

Théorème 1 (Equivalence [9]) *Les catégories ($* \to$ **Fonctions**$)[\mathcal{G}]$ et* **Co-injections**$[\mathcal{G}]$ *sont équivalentes pour les signatures de graphes hiérarchiques.*

2.6 Equivalence avec l'approche à double-pushout

L'approche par double-pushout, historiquement antérieure[6] définit une règle (et une transformation) comme une couple de flèches ayant même source. En outre l'approche par double-pushout ne travaille que sur la signature Graphes où les graphes étiquetés.

Le "collage" se faisant à partir de l'extrémité gauche de ce diagramme, on est obligé de créer une nouvelle construction appelée *Complément du pushout* (Cpo) pour arriver à dériver une règle en transformation. Ce complément n'existe que si deux conditions sont réalisées [5]:

Liaison Aucune arète $e \in G_{\text{Arète}} - m_{\text{Arète}}(L_{Arète})$ n'est incidente à un noeud $v \in m_{\text{Noeud}}((L_{Noeud} - l_{Noeud}(K_{Noeud}))$

Identification Il n'y a pas d'éléments $x, y \in L_{\text{Noeud}}$(resp.$L_{\text{Arète}}$)tels que $m_{\text{Noeud}}(x) = m_{\text{Noeud}}(y)$ et $y \notin K_{\text{Noeud}}$ (resp. tels que $m_{\text{Arète}}(x) = m_{\text{Arète}}(y)$ et $y \notin K_{\text{Arète}}$).

En outre il est unique si la partie gauche du couple est une injection [5] or dans ce cas là la paire est équivalente à un morphisme partiel [4]. Une paire de flèche de source commune avec une injection à gauche est appelée un $\mathcal{G}$-span.

Définition 9 (Equivalence des règles SPO et DPO) *Si $p : L \to R$ est une $\mathcal{G}$-transformation, $D(p) \stackrel{def}{=} \langle l : R_p \to L, p_{|R_p} : R_p \to R\rangle$ est le $\mathcal{G}$-span associé. Symétriquement, si $p = \langle l : K \to L, r : K \to R\rangle$ est un $\mathcal{G}$-span alors $S(p) : L \to R \stackrel{def}{=} r \circ l^{-1}$ est la $\mathcal{G}$-transformation associée avec $R_{S(p)} = l(K)$. On a $S(D(p)) = p$.*

On remarque que les deux conditions peuvent être étendues dans le contexte d'une signature quelconque et a quelques déductions près elles sont équivalents aux conditions (i) (pour l'identification) et (c) (pour la liaison). La propriété suivante achève la correspondance

Propriété 2 (Correspondance des dérivations [9,3]) *Soit $p : L \to R$ une $\mathcal{G}$-transformation, $D(p)$ son $\mathcal{G}$-span associé et m une ci-application. Les deux assertions suivantes sont équivalentes:*

- *$p_m : G \to H$ est la ci-dérivation de p selon m.*

– $D(p_m)$ *est le* $\mathcal{G}$*-span dérivé de* $D(p)$ *selon* m

Nous empruntons la conclusion de [3]. L'approche par simple-pushout est *complète* (une dérivation est toujours possible pour une règle et son collage), l'approche par double pushout est *reversible* (si une règle est dérivable, la dérivation de la règle prise en inversant L et R est l'inverse de la dérivation de cette règle). Ces deux propriétés sont incompatibles, la plupart des preuves que nous avons réalisées sur les grammaires de graphes reposent sur la réversibilité [8], voila pourquoi nous travaillons dans le contexte où le single-pushout est réversible et donc muni d'une condition sur les collages (i.e. ci).

3 Une approche reflective

Comme on l'a vu, l'association graphe/catégorie est structurelle dans cette théorie. L'autre intérêt de cette théorie et de pouvoir "transporter" des théories entière vers d'autres par l'utilisation de foncteurs. Notre contribution est un pas vers l'utilisation de ces deux caractéristiques de la théorie des catégories dans les grammaires de graphes.

La remarque fondamentale consiste à considérer qu'une signature S est un graphe dirigé donc un support de diagramme (le graphe sans l'étiquetage), et que l'étiquetage consistant d'un tel diagramme dans la catégorie **Set** (ou **Fonctions**) donne une S-algèbre. Ce point de vue permet d'envisager une généralisation de la notion d'algèbre en prenant comme catégorie d'interprétation une autre sous-catégorie de **Relation**. Dans la suite nous nous appuyerons sur la propriété suivante qui exprime cette intuition.

Propriété 3 (Reflection) *A toute Graphes-Algèbre A on peut associer une unique signature de graphe* $\texttt{Sign}(A)$.

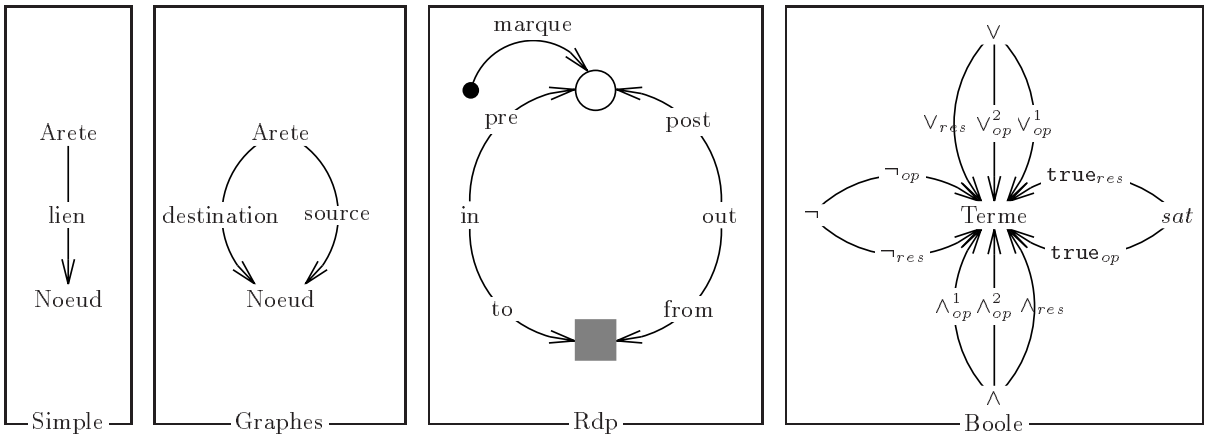


FIG. 5 –. *Signature de graphes comme des diagrammes.*

La première conséquence est qu'une signature peut se représenter de façon graphique. Nous en avons donné quelques exemples dans le figure 5. La première est une des signature hiérarchique les plus simples, la seconde est la signature des graphes dirigés, la troisième permet de représenter les réseaux de Petri marqués, la dernière sert à la représentation graphique de termes de logique booléenne.

Un autre avantage est de simplifier certaines définitions en adoptant des notations à base de fàches pour les opérateurs, ce qui permet de se passer des fonctions α et ω qui compliquent

la lecture des constructions (Cf. section 2.4). Cette simplification est particulièrement patente pour les signature de graphes hiérarchiques.

Définition 10 (Graphes Hiérarchiques) *Une signature $\mathcal{G}$ est une signature de graphe hiérarchique[9] ssi* $\mathtt{Sign}(\mathcal{G})^{-1}$ *est acyclique.*

La simplification réside ici dans l'utilisation d'un notion graphique pour décrire une propriété d'une signature. Mais le principal avantage réside dans l'extension réalisée sur une construction permettant de typer les graphes dirigés. Les graphes typés [4] sont le résultat d'une *construction* astucieuse [4]basée sur une catégorie comma: (**Fonctions**[*Graphes*] $\rightarrow$ **TG**) où TG (i.e une Graphes-algèbre) est le graphe typant. Le typage s'obtient par le morphisme induit par la catégorie comma du graphe typé vers le graphe typant. On peut caractériser les graphes typés dans notre approche:

Définition 11 (Graphes typés) *Une signature Σest une signature de graphe typé ssi il existe un Graphes-collage* $T : \mathtt{Sign}(\Sigma)^{-1} \rightarrow \mathtt{Sign}(Graphes)^{-1}$.

Pour achever l'inclusion on peut *construire* une signature TG pour laquelle **Fonctions**[TG] est équivalente à (**Fonctions**[*Graphes*] $\rightarrow$ **TG**). Cette construction "expense" chaque arête en une sorte et deux arêtes afférentes de cette sorte vers la source et la destination de l'arête originale. Ce motif étant le motif fondamental de la signature Graphes. L'équivalence des catégories étant une conséquence des propriétés des morphismes impliqués. On peut écrire la propriété suivante avec les classes de signature que l'on a définies: *Un graphe typé est hiérarchique.* C'est une conséquence du fait que toute sorte de type "Noeud" est sans successeur et que toute sorte de type "Arete" a deux successeurs de type "Noeud". La réciproque est fausse, considérer Rdp (exemple de surchage en considérant un jeton comme une boucle) et pour en finir Boole que l'on ne peut surcharger sans changer de façon importante la structure (les opérateurs devenant d'arité quelconque). Ceci montre que notre approche étend l'approche de [4]. Un jeton est ici un attribut de la place auquel il est attaché.

Une conséquence importante de ce plongement est que le mérite des constructions de [4] ne s'arrête pas là, il montre qu'un collage de TG dans TG' induit un foncteur préservant les dérivations. L'approche est importable dans notre démarche. Pour les collage qui sont des injections, la démonstration est assez simple dans notre approche car l'ensemble des constructions utilisées sont faites par induction sur la structure de la signature (chaque sorte puis chaque opérateur indépendemment), et donc si on a inclusion des signature, le résultat reste le même pour la partie concernée par l'inclusion. Donc, on peut voir un système ayant une signature complexe à travers cette traduction de sa structure et cette simplification est compatible avec la traduction du système de règles. De telles traductions simplifiantes et cohérentes (i.e. des foncteurs) sont très utiles pour dépasser la limitation des approches à visualisation intégrée: le syndrome du "sac de noeud".

Conclusion et travaux en cours

Ce travail a présenté la dérivation de transformation de graphes dans l'approche par simple pushout. Nous avons montré que cette approche traitait explicitement le problème rencontré dès que l'effectif des composants évolue dans le temps: l'état du système dépend de la politique d'allocation des références. En outre dans cette approche les collages, les transformations et les équivalences sont un seul et même objet, un morphisme, paramétré par le type de ses composantes (fonctions, co-injections ou bijections). La construction fondamentale permettant la dérivation: le pushout, peut être décomposé en deux opérations[11] dont l'une recouvre exactement la notion de politique d'allocation. L'état du système est lui représenté par une algèbre

sur une signature unaire[9]. Cette extension permet de batir une théorie des grammaires de graphes dont la complexité des objets représentée est variable à loisir suivant les besoins. En outre elle permet de doter les noeuds d'attributs et de différencier ces attributs des boucles dans la topologie reliant les noeuds. En outre une telle signature est syntaxiquement un graphe dirigé et on peut donc associer de façon univoque une signature à toute algèbre de la signature Graphes. Une piste forte de promesses et de considérer des morphismes de signatures. Cela permet de définir des vues formelles (i.e. des foncteurs) entre des structures différentes. On peut ainsi observer le déroulement d'un système à travers plusieurs vues cohérentes qui sont d'une compléxité raisonnable pour être visualisées.

Pour valider l'approche générale que nous venons de décrire, un prototype ML doté d'une interface graphique performante est actuellement en cours de développement. Cette implémentation s'appuie sur les travaux de [11]. Donc nous avons choisi une politique (par maximum) et développé en suivant le cheminement de la deuxième partie un système de réécriture de graphes à signature variable. L'avantage principal étant la possibilité de réaliser dans le prototype des conversions d'une signature à l'autre et d'importer des propriétés. Pour l'instant l'application de ce type concerne un moteur de construction de "workflow" où la signature exacte utilisée (proche de Boole) est beaucoup plus précise qu'un codage en graphe étiqueté, et la transcription en Réseaux de Petri est automatisée. Ceci permet de compléter le travail de [1] en *démontrant* l'équivalence entre des règles de transformation de réseaux de Petri et celle des "workflow" et d'importer des propriétés structurellement décidables des réseaux de Petri (ici borné, vivant). Le prototype réalise ces conversions d'une signature à l'autre. La suite de ce travail doit permettre de construire des signatures et des systèmes de règles et de suivre leur déroulement ou leur analyse suivant différentes vues simplifiées et cohérentes.

Références

1. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, Berlin, 1997.
2. Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, 1998.
3. Andrea Corradini, Hartmut Ehrig, Reiko Heckel, Martin Korff, Michael Löwe, Leila Ribeiro, and Anika Wagner. Algebraic approaches to graph transformation - part I: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, chapter 4, pages 247–312. World Scientific, 1997.
4. Andrea Corradini and Reiko Heckel. A compositional approach to structuring and refinement of typed graph grammars. In A. Corradini and U. Montanari, editors, *SEGRAGRA'95, Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation*, volume 2. Elsevier, 1995.
5. Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation - part I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, chapter 3, pages 163–246. World Scientific, 1997.
6. H. Ehrig. Tutorial introduction to the algebraic approach of graph-grammars. In A. Rosenfeld H. Ehrig, M. Nagl, G. Rozenberg, editor, *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 291 of *LNCS*, pages 3–14, Warrenton, VA, December 1986. Springer Verlag.
7. H. Ehrig and B. Mahr. Fundamentals of Algebraic Specifications. In *Monographs on Theoretical Computer Science 1 (2)*, volume Volume 6 (21) of *EATCS*. Springer-Verlag, 1985 (1990).
8. Pierre Gradit and François Vernadat. Les grammaires de reécriture de graphes pour la vérification d'algorithmes distribués. In *NOuvelles TEchnologies de la REpartition '97, Pau*, November 1997.
9. Michael Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109(1–2):181–224, March 1993.
10. Saunders MacLane. *Categories for Working Mathematicians.* Springer-Verlag, New York, 1971.
11. D. E. Rydeheard and R. M. Burstall. *Computational Category Theory.* Prentice-Hall, New York, 1988. ISBN 0-13-162736-8.