DIAGRAMMES

C. ORIAT

Étude des spécifications modulaires : constructions de colimites finies, diagrammes, isomorphismes. Partie II

Diagrammes, tome 40 (1998), p. 3-71

<http://www.numdam.org/item?id=DIA_1998__40__3_0>

DIAGRAMMES VOLUME 40 , 1998

ETUDE DES SPECIFICATIONS MODULAIRES : CONSTRUCTIONS DE COLIMITES FINIES, DIAGRAMMES, ISOMORPHISMES (*)

PARTIE II (**)

C. Oriat

RESUME La composition de spécifications modulaires peut être modélisée, dans le formalisme des catégories, par des colimites de diagrammes. La somme amalgamée permet en particulier d'assembler deux spécifications en précisant les parties communes. Notre travail poursuit cette idée classique selon trois axes

D'un point de vue syntaxique, nous définissons un langage pour représenter les spécifications modulaires construites à partir d'une catégorie de spécifications et de morphismes de spécifications de base Ce langage est caractérisé formellement par une catégorie de termes finiment cocomplète.

D'un point de vue sémantique, nous proposons d'associer à tout terme un diagramme. Cette interprétation permet de faire abstraction de certains choix effectués lors de la construction de la spécification modulaire. Pour cela, nous définissons une catégorie de diagrammes "concrète", c'est-à-dire dont les flèches peuvent être manipulées effectivement. En considérant le quotient par une certaine congruence, nous obtenons une complétion de la catégorie de base par colimites finies. Nous montrons que le calcul du diagramme associé à un terme définit une équivalence entre la catégorie des termes et la catégorie des diagrammes, ce qui prouve la correction de cette interprétation.

Enfin, nous proposons un algorithme pour décider si deux diagrammes sont isomorphes, dans le cas particulier où la catégorie de base est finie et sans cycle. Cela permet de détecter des isomorphismes "de construction" entre spécifications modulaires, c'est-à-dire des isomorphismes qui ne dépendent pas des spécifications de base, mais seulement de la manière dont celles-ci sont assemblées.

ABSTRACT. The composition of modular specifications can be modeled, in a categoy theoretic framework, by means of colimits of diagrams. Pushouts in particular allow us to

---

(*) Thèse d'Informatique, préparée et soutenue à l'Institut National Polytechnique de Grenoble (Laboratoire LSR-IMAG).

(**) La Partie I de ce travail est publiée dans Diagrammes 39 et la Partie III le sera dans Diagrammes 41.

gather two specifications sharing a common part. Our work extends this classic idea along three lines.

From a syntactic point of view, we define a language to represent modular specifications built from a category of base specifications and base specification morphisms. This language is formally characterized by a finitely cocomplete category of terms.

From a semantic point of view, we propose to associate with each term a diagram. This interpretation allows us to abstract some choices made while constructing a modular specification. We thus define a "concrete" category of diagrams, in which arrows can actually be handled. Considering the quotient by a certain congruence relation, we get a completion of the base category with finite colimits. We prove that this calculus defines an equivalence between the category of terms and the category of diagrams, which shows the soundness of this interpretation.

At last, we propose an algorithm to decide whether two diagrams are isomorphic, when the base category is finite and cycle free. This allows us to detect "construction isomorphisms" between modular specifications, i.e. isomorphisms which do not depend on the base specifications, but only on their combination.

Introduction

Spécifier un problème consiste à le décrire sans décider prématurément de la façon dont il sera résolu. En informatique, une spécification de programme est une description des fonctionnalités attendues du programme indépendante des choix de mise en œuvre.

Il existe plusieurs manières de spécifier un programme. On peut utiliser une langue naturelle, par exemple en écrivant un cahier des charges. Certaines activités, comme la validation de programme, la construction automatique d'un prototype, ou la production systématique de jeux de tests, nécessitent des spécifications *formelles*, qui possèdent une sémantique précise, en particulier non ambiguë.

Nous nous intéressons ici aux *spécifications algébriques*, qui font partie des spécifications *orientées propriétés*[1]. Les spécifications orientées propriétés sont basées sur un *système logique*, qui définit l'ensemble des propriétés que doit satisfaire tout modèle de la spécification.

1 Spécifications algébriques

Une spécification algébrique est composée d'une signature et d'un ensemble d'axiomes. La signature contient les symboles utilisés pour former des expressions, et les axiomes définissent certaines propriétés de ces expressions. La logique sous-jacente décrit l'ensemble des *théorèmes* que l'on peut déduire des axiomes. De plus, la *sémantique* des spécifications algébriques associe à toute spécification une classe de *modèles*, c'est-à-dire une classe d'algèbres multi-sortes qui satisfont les axiomes.

Historiquement, les spécifications algébriques sont issues de deux courants distincts : l'algèbre universelle en mathématiques [Coh65] et les types abstraits en génie logiciel. L'origine des types abstraits figure dans le concept de *classe* du langage de programmation Simula [DN66, BDMN73]. Un type abstrait, composé d'un ensemble de *domaines de valeurs* accompagné d'un ensemble d'*opérations* sur ces domaines de valeurs, permet de structurer les données. L'idée de base en spécification algébrique consiste à modéliser un type abstrait par une algèbre. Cette idée est motivée par l'analogie entre un type abstrait et une algèbre multi-sortes sur une signature. L'interprétation des sortes de la signature correspond en effet aux domaines de valeurs, et l'interprétation des opérateurs de la signature correspond aux opérations sur ces domaines de valeurs.

Les premiers travaux sur les spécifications algébriques datent du milieu des années 1970, avec les travaux de B. Liskov et S. Zilles [LZ74], de J. Guttag [Gut75,

1. En anglais, *property-oriented.*

Gut77] et du groupe ADJ (composé de J. Goguen, J. Thatcher, E. Wagner et J. Wright) [GTWW75, GTWW77, GTW78]. Au départ, les spécifications algébriques étaient basées sur la logique équationnelle. Puis ont été introduites d'autres logiques, comme les clauses de Horn, la logique du premier ordre ou encore les spécifications avec contraintes [EM90]. D'autres extensions ont permis de prendre en compte la notion d'erreur [BBC86, BL93]. J. Goguen et R. Burstall ont proposé la *théorie des institutions*, qui offre un cadre général pour étudier les différents formalismes de spécification algébrique [GB84, GB90]. Il existe beaucoup de références sur les spécifications algébriques, parmi les bonnes synthèses, citons [EM85, MG85, Wir90].

Ces travaux ont donné naissance à de nombreux langages de spécification algébrique, comme Clear [BG77, BG80], ACT ONE et ACT TWO [EM85, EM90], ASL [SW83, Wir86], OBJ2 et OBJ3 [FGJM85, GKK+87], PLUSS [Gau84, Bid89], LPG [Ber83, BE86, B+90], GLIDER [Huf92]. On trouve un bon panorama des différents langages de spécification algébrique existants dans [Wir94].

2 Théorie des catégories

La théorie des catégories repose sur deux notions : l'*objet* et la *flèche*, contrairement à la théorie des ensembles qui repose sur le seul concept d'*ensemble*. En théorie des ensembles, un ensemble est caractérisé de façon *interne* par ses *éléments*, alors qu'en théorie des catégories, un *objet* est caractérisé de façon *externe* par les relations entretenues par l'intermédiaire des flèches avec les autres objets. Pour cette raison, un grand nombre de concepts de théorie des catégories sont définis *à un isomorphisme près*.

La théorie des catégories a été inventée au début des années 1940 par S. Mac Lane et S. Eilenberg. Bien que certains mathématiciens aient douté de l'intérêt des catégories, ce formalisme a joué un rôle unificateur en mathématiques, en particulier en algèbre et en topologie. Cette théorie définit les différents concepts de façon générale et abstraite, indépendamment de tout modèle. (C'est pour cette raison que ses détracteurs l'on surnommée "non-sens abstrait[2]".) La théorie des catégories a commencé à intéresser les informaticiens au début des années 1970. Par exemple, la découverte de l'équivalence entre le λ-calcul typé et les catégories cartésiennes fermées a permis de donner des solutions élégantes au problème des modèles du λ-calcul, lié à celui de la résolution des équations de domaines [SP77, Wan79, LS86]. Il existe de façon générale des liens très étroits entre la théorie des types et la théorie des catégories (cf. par exemple [See84, Poi92]).

L'ouvrage de référence sur les catégories est le livre de S. Mac Lane [McL71], plutôt destiné aux mathématiciens. Pour les informaticiens, le livre de M. Barr et C. Wells [BW90] est plus accessible, car les différents concepts sont présentés de façon assez élémentaire et les exemples sont tirés de l'informatique. Il existe beaucoup d'autres ouvrages qui traitent des catégories, par exemple [MB70, AM75, Gol79, AL91].

Le formalisme des spécifications algébriques s'appuie largement sur la théorie

2. En anglais, *abstract nonsense*.

des catégories. Historiquement, le développement des spécifications algébriques (en particulier les travaux du groupe ADJ) a été influencé par les théories algébriques de Lawvere. Les théories algébriques de Lawvere permettent de modéliser les catégories d'algèbres en faisant abstraction de la syntaxe, c'est-à-dire indépendamment de la notion de signature [Law63, BW85, WBT85].

D'autre part, d'un point de vue pratique, on peut justifier l'utilisation des catégories par l'importance des *morphismes de spécifications*, qui jouent un rôle essentiel en spécification algébrique. Intuitivement, un morphisme de spécifications exprime une relation entre deux spécifications. Pour prendre un exemple mathématique, tout anneau peut être considéré comme un groupe, ce qui signifie qu'il existe un morphisme de spécifications entre la spécification des groupes et la spécification des anneaux. Il peut exister plusieurs morphismes de spécifications entre deux spécifications. Par exemple, on peut considérer tout anneau comme un monoïde *de deux façons différentes*, en considérant soit l'opérateur additif, soit l'opérateur multiplicatif de l'anneau comme l'opérateur du monoïde. Préciser quel opérateur de l'anneau sera l'opérateur du monoïde consiste exactement à définir un morphisme de spécifications entre la spécification des monoïdes et la spécification des anneaux.

3 Modularité

Le développement de spécifications de grande taille nécessite un découpage des spécifications en plusieurs spécifications plus simples, appelées *modules*. Chaque module correspond à un problème plus simple à résoudre. Cette méthode, qui consiste à "diviser pour mieux régner" est classique en génie logiciel [LCW85]. De façon générale, la modularité permet d'une part le développement indépendant des différents modules, et d'autre part une meilleure compréhension de chaque module, ce qui facilite la maintenance.

Cette vision de la modularité correspond à une approche descendante de résolution de problème, puisque l'accent est mis sur la *décomposition* du problème en plusieurs sous-problèmes plus simples. Dans notre travail, notre conception de la modularité est au contraire ascendante. Nous nous intéressons en effet à la *composition* des modules, c'est-à-dire à l'assemblage de modules élémentaires permettant de construire des modules plus complexes. Ces modules élémentaires sont regroupés dans une bibliothèque, la bibliothèque des *modules de base*. L'intérêt de la composition est de permettre la *réutilisation* des modules de base, préconisée en génie logiciel pour réduire les erreurs et les coûts de développement (cf. par exemple [Ber90]).

Un aspect de la modularité est la possibilité de définir des spécifications *génériques*, c'est-à-dire paramétrées [SST92]. En effet, une spécification paramétrée peut être instanciée de différentes manières par d'autres spécifications, ce qui évite de réécrire une nouvelle spécification pour chaque variation d'un même problème. Le but est alors de définir des spécifications génériques suffisamment générales pour permettre de spécifier le maximum de problèmes par une simple instanciation [Mar95].

Le développement modulaire des spécifications est assisté dans les langages de spécification algébrique par des *opérateurs de construction*[3] comme le *renommage*,

3. En anglais, *specification-building operations*.

l'*enrichissement* ou l'*abstraction* [ST88, Wir90]. Ces primitives permettent de construire de nouvelles spécifications à partir de spécifications déjà définies. La composition de modules est un cas particulier d'opérateur de construction de spécifications, qui peut être modélisé par des *colimites de diagrammes*. Intuitivement, en théorie des catégories, un *diagramme* permet de décrire un assemblage de plusieurs objets en précisant des *partages* entre certains objets à l'aide de flèches. La *colimite* du diagramme permet de considérer le résultat de cette composition. Cette utilisation générale des colimites pour modéliser l'interconnexion de systèmes a été proposée par J. Goguen [Gog73, Gog92].

Un cas particulier de colimite est la *somme amalgamée*[4], qui correspond à l'assemblage de deux objets B et C, en spécifiant une partie commune entre ces deux objets par un *objet partagé* A et deux flèches $f : A \to B$ et $g : A \to C$. En spécification algébrique, la somme amalgamée peut modéliser d'une part la *composition* de deux spécifications qui ont une partie commune, et d'autre part l'*instanciation* d'une spécification générique.

4 Notre travail

Nous étudions la composition des spécifications algébriques modulaires. Nous supposons que nous disposons d'une bibliothèque de spécifications et de morphismes de spécifications *de base*, qui forment une catégorie *de base*, en général appelée $\mathcal{C}_0$ dans ce mémoire. Les spécifications de base sont monolithiques, c'est-à-dire non décomposées. Notre travail repose sur une idée classique en spécification algébrique : la composition des spécifications peut être modélisée par des colimites de diagrammes finis. Nous proposons donc de considérer une *spécification modulaire* comme une spécification obtenue à partir de spécifications et morphismes de spécifications de base par une suite de constructions de colimites.

Notre travail est indépendant de la logique sous-jacente du langage de spécification algébrique. Nous pouvons donc nous placer dans le cadre général des *institutions*. Nous devons uniquement supposer que la catégorie des spécifications est *finiment cocomplète*, c'est-à-dire que tout diagramme fini a une colimite.

D'un point de vue syntaxique, c'est-à-dire en ce qui concerne le *langage* utilisé pour décrire ces constructions, nous ne considérons pas toutes les colimites finies, mais uniquement les sommes amalgamées. En effet, avec la spécification vide, objet initial de la catégorie des spécifications, les sommes amalgamées permettent de simuler la construction de toute colimite finie. Nous proposons la construction d'une catégorie de *termes*, appelée Terme($\mathcal{C}_0$), pour représenter la composition de spécifications modulaires en utilisant les sommes amalgamées.

En théorie des catégories, on travaille le plus souvent à un isomorphisme près. C'est une conséquence de la caractérisation externe des objets dans ce formalisme. Les colimites, par exemple, sont définies au départ à un isomorphisme près. La définition d'une syntaxe est au contraire un problème typiquement informatique, qui nécessite de faire des *choix de représentation*. C'est la raison pour laquelle nous devons utiliser des *sommes amalgamées choisies* au moment de la définition du langage

4. En anglais, *pushout*. Les catégoriciens utilisent également le terme de *coproduit fibré*.

d'expression pour les spécifications modulaires. Ces choix de sommes amalgamées correspondent à des choix de codage.

Un terme correspond donc à une spécification modulaire construite par une succession de sommes amalgamées. Dans une seconde étape, nous proposons d'interpréter un terme par un *diagramme* dont la colimite est la spécification modulaire dénotées par ce terme. Nous considérons les diagrammes comme une sémantique pour les spécifications modulaires, par opposition aux termes qui constituent la syntaxe.

L'interprétation des termes par des diagrammes nécessite de définir une *catégorie* de diagrammes puisqu'il faut non seulement associer à toute spécification modulaire un diagramme, mais aussi associer à tout morphisme de spécifications modulaires un morphisme de diagrammes. La définition d'un diagramme est relativement standard en théorie des catégories, bien qu'il existe quelques nuances. La définition de morphisme de diagrammes est moins connue, en particulier dans la littérature informatique où sont utilisées des définitions moins générales que celle dont nous nous servons ici.

Nous présentons deux catégories de diagrammes, DIAGR($\mathcal{C}_0$) et diagr($\mathcal{C}_0$). Les objets de DIAGR($\mathcal{C}_0$) sont des *diagrammes*, et les flèches de DIAGR($\mathcal{C}_0$) des *morphismes de diagrammes*. La catégorie diagr($\mathcal{C}_0$) est obtenue à partir de DIAGR($\mathcal{C}_0$) par un quotient par une relation de congruence sur les flèches de DIAGR($\mathcal{C}_0$). C'est la catégorie diagr($\mathcal{C}_0$) qui possède les propriétés théoriques intéressantes. Il s'agit en effet d'une catégorie *finiment cocomplète*, c'est-à-dire que tout diagramme fini sur diagr($\mathcal{C}_0$) a une colimite finie dans diagr($\mathcal{C}_0$). D'autre part, diagr($\mathcal{C}_0$) est une complétion par colimites finies de la catégorie $\mathcal{C}_0$. Sur le plan pratique, comme les flèches de diagr($\mathcal{C}_0$) sont des classes d'équivalence de morphismes de diagrammes, on travaille en réalité avec des représentants, c'est-à-dire avec des flèches de DIAGR($\mathcal{C}_0$). La catégorie DIAGR($\mathcal{C}_0$) permet donc la manipulation effective des flèches de diagr($\mathcal{C}_0$).

La représentation des termes par des diagrammes est définie par un foncteur

$$\mathcal{D} : \text{Terme}(\mathcal{C}_0) \to \text{diagr}(\mathcal{C}_0)$$

entre les catégories Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$). Nous montrons que ce foncteur définit une *équivalence de catégories* entre Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$), ce qui signifie que bien qu'elles ne soient pas isomorphes, ces catégories ont néanmoins essentiellement la même structure.

Après avoir représenté les spécifications modulaires par des diagrammes, nous nous intéressons au problème de l'*équivalence* entre deux spécifications modulaires. Deux spécifications modulaires sont considérées comme équivalentes lorsque celles-ci sont *isomorphes* en tant que *colimites de diagramme*. Détecter cet isomorphisme consiste en fait à détecter un isomorphisme entre deux diagrammes dans la catégorie diagr($\mathcal{C}_0$). Nous appelons cet isomorphisme un *isomorphisme de construction*, parce qu'il ne dépend pas de la définition effective des spécifications de base, mais seulement de la manière dont celles-ci sont combinées pour construire les spécifications modulaires. Dans l'hypothèse où la catégorie $\mathcal{C}_0$ est finie et ne comporte pas de cycle, nous proposons un algorithme pour détecter si deux diagrammes sont isomorphes. Des résultats partiels concernant l'interprétation des termes par des diagrammes et la détection d'isomorphismes de construction ont été présentés dans [Ori94, Ori95].

5 Comparaisons avec d'autres travaux

Catégories de diagrammes

La définition de *diagramme* est, à quelques nuances près, standard. Chez S. Mac Lane [McL71], un diagramme sur une catégorie $\mathcal{C}$ est un *foncteur* d'une catégorie J vers $\mathcal{C}$. La définition de M. Barr et C. Wells [BW90], est moins générale, puisqu'un diagramme est un *morphisme de graphes* d'un graphe vers le graphe sous-jacent de $\mathcal{C}$. Nous utilisons une définition "intermédiaire" : pour nous, un diagramme est un *foncteur* d'une catégorie *librement engendrée sur un graphe* vers $\mathcal{C}$. Partir d'un graphe nous permet de rester proche de l'informatique. Par contre, nous ne souhaitons pas nous restreindre à une *petite* catégorie $\mathcal{C}$.

La définition de *morphisme de diagrammes* est beaucoup moins standard. Par exemple dans Clear, un morphisme de diagrammes, appelé *morphisme basé*, correspond uniquement à une *inclusion* entre deux diagrammes.

La définition proposée par A. Tarlecki, R. Burstall et J. Goguen dans [TBG91], bien que plus générale que celle utilisée dans la sémantique du langage Clear, n'est pas encore suffisamment générale pour notre travail. En effet, cette définition ne permet pas de représenter tout morphisme de spécifications modulaires par un morphisme de diagrammes.

La définition de morphisme de diagrammes que nous utilisons n'est pas nouvelle en théorie des catégories. Une version duale de la catégorie $\mathrm{diagr}(\mathcal{C}_0)$ est par exemple utilisée par M. Barr et C. Wells dans [BW94]. Néanmoins, la reformulation de la définition de catégorie de diagrammes que nous proposons dans le chapitre 2 présente un intérêt en informatique, d'une part parce que la catégorie $\mathrm{DIAGR}(\mathcal{C}_0)$ permet de manipuler effectivement les morphismes de diagrammes, et d'autre part parce que les catégories $\mathrm{DIAGR}(\mathcal{C}_0)$ et $\mathrm{diagr}(\mathcal{C}_0)$ sont peu connues dans cette discipline.

Calculs de colimites

R. Burstall et D. Rydeheard proposent des algorithmes pour *calculer* certains concepts de théorie des catégories [Bur80, BR86], en particulier les colimites de diagrammes. Le calcul général de colimite est paramétré par certaines colimites, en l'occurrence un objet initial, des sommes et des coégalisateurs, qui permettent d'évaluer la colimite d'un diagramme quelconque. Ainsi, ayant défini l'objet initial, la somme et le coégalisateur par exemple dans la catégorie des ensembles **Set**, on en déduit un moyen de calculer la colimite d'un diagramme quelconque dans **Set**. Cet algorithme est en fait basé sur une preuve du résultat suivant : si une catégorie a un objet initial, des sommes et des coégalisateurs, alors celle-ci est finiment cocomplète. Notre travail est basé sur un théorème similaire : si une catégorie a un objet initial et des sommes amalgamées alors celle-ci est finiment cocomplète. Notre travail est différent, dans la mesure où nous ne cherchons pas à calculer des colimites dans une catégorie fixée. En particulier, nous nous intéressons au problème général de l'isomorphisme entre deux diagrammes, qui correspond à un isomorphisme entre leurs colimites respectives. Cet isomorphisme est indépendant de la catégorie sur laquelle ces diagrammes sont construits.

Utilisation des colimites en spécification algébrique

L'utilisation des colimites pour représenter l'assemblage de spécifications est loin d'être nouvelle. R. Burstall et J. Goguen ont présenté cette idée dans la sémantique du langage de spécification Clear [BG80]. Dans Clear, la composition de plusieurs spécifications qui partagent un environnement commun, appelées *théories basées*, est en effet formalisée par la colimite d'un diagramme.

Les colimites ont ensuite été assez intensivement utilisées en spécification algébrique. La somme amalgamée en particulier permet de modéliser l'instanciation de spécifications génériques [TWW82, EM85]. H. Ehrig, R. Jimenez et F. Orejas [EJO93] ont également présenté une forme plus générale d'instanciation à l'aide de *sommes amalgamées multiples*[5], nouvel exemple de colimite.

Syntaxe pour les constructions modulaires

J. Bergstra, J. Heering et R. Klint [BHK90] d'une part, et G. Renardel de Lavalette [Ren91] d'autre part, ont proposé de structurer les opérateurs de construction de spécifications sous forme d'*algèbres de modules*. Une algèbre de modules est un *langage* permettant d'écrire des *expressions* de composition des spécifications algébriques. Cette approche est voisine de notre travail sur le langage Terme($\mathcal{C}_0$), puisque le problème de la composition des spécifications algébriques est abordé d'un point de vue *syntaxique*, et permet de comparer différentes spécifications modulaires. Par contre, nous ne nous intéressons pas aux mêmes opérateurs de constructions de spécifications, puisque nous ne considérons que les constructions de colimites.

Notre syntaxe pour représenter les spécifications algébriques modulaires est largement inspirée des travaux de J.-C. Reynaud, qui a proposé un système de types pour représenter les constructions de colimites [Rey90a, Rey90b, Rey93]. Ce système est fondé sur les théories algébriques généralisées de Cartmell [Car86] qui permettent de spécifier des types dépendants, c'est-à-dire des types paramétrés par des termes. Les types dépendants ont été également utilisés par T. Streicher et M. Wirsing [SW91] pour modéliser la composition de spécifications algébriques modulaires. D'autre part, H. Ehrig, R. Jimenez et F. Orejas ont également proposé une syntaxe pour les spécifications algébriques modulaires dans [EJO93]. Cette syntaxe est en partie basée sur des constructions de sommes amalgamées.

La principale difficulté en ce qui concerne la définition d'une syntaxe pour les constructions de colimites est le problème de circularité entre la définition d'une part des objets et des flèches de Terme($\mathcal{C}_0$) et d'autre part de l'égalité entre deux flèches de Terme($\mathcal{C}_0$). En effet, il existe une flèche, que nous appelons **up** dans ce mémoire, entre une somme amalgamée et un objet *à condition qu'une égalité entre deux flèches de* Terme($\mathcal{C}_0$) *soit vérifiée*. La définition des flèches **up** dépend donc de la définition de l'égalité des flèches dans Terme($\mathcal{C}_0$), qui elle-même présuppose que les flèches ont été définies.

J.-C. Reynaud contourne le problème en proposant une définition *concrète*, c'est-à-dire sémantique, d'une catégorie librement engendrée par objet initial choisi et sommes amalgamées choisies [Rey93]. H. Ehrig, R. Jimenez et F. Orejas [EJO93] ne

5. En anglais, *multiple pushout*.

définissent pas de syntaxe pour les flèches **up**. La syntaxe qu'ils proposent ne permet donc pas d'obtenir une catégorie *finiment cocomplète.*

Une solution pour spécifier une catégorie finiment cocomplète sans utiliser d'égalités pour définir les flèches a été présentée par F. Cury dans [Cur91]. F. Cury propose de dédoubler les flèches **up** en deux flèches dont la définition ne dépend pas d'une égalité. Ces deux flèches, qui sont des exemples particuliers de flèches **up**, permettent de reconstruire *a posteriori* toutes les flèches **up**.

La solution que nous proposons consiste à stratifier la construction de Terme($\mathcal{C}_0$) en une suite de catégories $\mathcal{C}_i$. L'existence d'une flèche **up** dans la catégorie $\mathcal{C}_i$ dépend uniquement d'égalités dans la catégorie $\mathcal{C}_{i-1}$. Nous évitons ainsi le problème de circularité entre la définition des flèches et la définition des égalités.

Notre catégorie Terme($\mathcal{C}_0$) peut être comparée au *type* d'une esquisse, introduit par C. Ehresmann [Ehr68]. Dans les travaux de D. Duval et J.-C. Reynaud [DR94a, DR94b], le type d'une esquisse est une *catégorie librement engendrée* à partir d'un graphe par *sommes* et *produits.* Par conséquent, la construction du type d'une esquisse est basée sur les catégories à sommes et produits alors que la construction de Terme($\mathcal{C}_0$) est basée sur les catégories à colimites finies. D'autre part, une esquisse contient des cônes et cocônes distingués, qui spécifient certaines limites ou colimites. Par contre, dans la construction de Terme($\mathcal{C}_0$) que nous proposons, il n'est pas possible de spécifier des sommes amalgamées distinguées dans la catégorie de base $\mathcal{C}_0$.

6 Plan de ce mémoire

Ce mémoire est divisé en trois parties, chaque partie faisant l'objet d'un volume de *Diagrammes.*

Dans la première partie, qui comprend les chapitres 1 et 2, nous avons présenté le cadre général de notre travail (les spécifications modulaires), ainsi que les bases théoriques (les catégories de diagrammes).

Ce volume contient la deuxième partie de notre thèse, qui correspond au chapitre 3. Ce chapitre est consacré à la syntaxe des spécifications modulaires. Nous proposons une construction stratifiée de la catégorie de termes Terme($\mathcal{C}_0$), qui fournit un langage pour représenter les spécifications et morphismes de spécifications modulaires. Nous montrons d'une part que Terme($\mathcal{C}_0$) est une catégorie finiment cocomplète, et d'autre part que Terme($\mathcal{C}_0$) est une extension conservatrice de $\mathcal{C}_0$. Enfin, nous construisons, à partir de Terme($\mathcal{C}_0$), une catégorie $\mathcal{L}(\mathcal{C}_0)$ librement engendrée par objet initial choisi et sommes amalgamées choisies sur $\mathcal{C}_0$.

La troisième partie, qui fait l'objet d'un troisième volume de *Diagrammes*, sera consacrée à l'interprétation des termes par des diagrammes (chapitre 4) et à la présentation d'un algorithme permettant de détecter si deux diagrammes sont isomorphes, dans le cas où la catégorie de base est finie et sans cycle (chapitre 5).

Chapitre 3

Syntaxe : catégorie des termes

> *"Nous sommes certain que la syntaxe est indispensable mais nous voulons soutenir qu'elle ne saurait être syntaxe que dans la mesure où elle envisage non pas des relations entre objets quelconques, mais des relations entre des "signes" ou des "expressions". Or ce n'est que comme concept pragmatique que nous pourrons aborder la notion de "signe" ou d'"expression", car un événement n'est un signe que s'il est émis ou reçu comme tel."*
> — *Léo Apostel, Logique et connaissance scientifique*

Dans ce chapitre, nous considérons une catégorie $\mathcal{C}_0$ *de base* dont les objets sont considérés comme "atomiques", c'est-à-dire ne peuvent pas être divisés. Nous supposons que cette catégorie $\mathcal{C}_0$ est *petite*. Nous souhaitons pouvoir construire des objets composites en réunissant ces objets élémentaires, en fusionnant éventuellement certaines parties communes.

L'assemblage de plusieurs objets est modélisé à l'aide de constructions de colimites. Par exemple, dans la catégorie des ensembles, la somme permet de modéliser l'union disjointe de deux ensembles. La somme amalgamée permet de réunir deux ensembles en fusionnant certains éléments. Dans le cadre des spécifications algébriques, la somme amalgamée permet également de combiner deux spécifications en précisant les parties fusionnées.

Le but de ce chapitre est de présenter une *syntaxe*, c'est-à-dire un langage de description des constructions de colimites. Ces constructions de colimites font partie d'une *catégorie*, parce qu'à partir des objets et flèches de $\mathcal{C}_0$, on peut non seulement construire des objets colimites, mais également des flèches entre objets colimites. En effet, d'un point de vue formel, la fonction qui à tout diagramme associe sa colimite peut s'appliquer sur les flèches et être ainsi étendue en un foncteur. Dans ce chapitre, nous construisons donc une catégorie de termes, appelée $\mathrm{Terme}(\mathcal{C}_0)$, dont les objets représentent des constructions de colimites.

Notre objectif n'est pas d'avoir une syntaxe pour *toutes* les constructions de coli-

mites finies, mais de sélectionner quelques constructions qui permettent d'obtenir un objet isomorphe à n'importe quelle colimite finie. Il s'agit donc de pouvoir décrire une catégorie finiment cocomplète. Ici, nous choisissons d'avoir une représentation pour un objet initial, et pour des sommes amalgamées. Ce choix est purement arbitraire, on aurait pu prendre d'autres colimites, comme des sommes ou des coégalisateurs.

Nous avons donc besoin de constructions syntaxiques pour l'objet initial et les sommes amalgamées. Nous utilisons le terme $\emptyset$ pour représenter l'objet initial, et le terme **push** (A, B, C, f, g) pour représenter la somme amalgamée construite sur le diagramme qui comporte trois nœuds étiquetés par A, B, et C, et deux flèches étiquetées par $f : A \to B$ et $g : A \to C$. Comme nos constructions sont purement syntaxiques, la seule égalité dont nous disposons, au départ, est *l'identité syntaxique*. Nous définissons donc des *relations* entre les flèches pour décrire les propriétés de l'objet initial et des sommes amalgamées. Techniquement, nous définissons donc une *précatégorie*, c'est-à-dire une catégorie dans laquelle certaines égalités sont remplacées par des équivalences sur les flèches. La catégorie correspondante est le quotient de cette précatégorie par la relation d'équivalence.

Comme l'existence de certains termes est conditionnée par une relation entre deux autres termes (cf. règle (10)), on ne peut pas définir successivement les termes, puis la relation sur les termes. Pour résoudre ce problème, nous proposons une construction stratifiée de la syntaxe. Nous définissons une suite de précatégories de termes $\mathcal{C}_i$, telles que, dans la précatégorie $\mathcal{C}_i$, l'introduction de termes n'est conditionnée que par des relations entre termes de $\mathcal{C}_{i-1}$. Finalement, la précatégorie des termes TERME($\mathcal{C}_0$) est la "limite" de la suite de précatégories $\mathcal{C}_i$.

Syntaxe, sémantique, catégories et choix

En informatique, les problèmes ont souvent deux aspects : un aspect *syntaxique*, qui correspond au langage qui décrit les objets manipulés (par exemple un programme) ; et un aspect sémantique, qui correspond au sens des objets manipulés (par exemple le résultat de l'exécution d'un programme). Du point de vue syntaxique, on fait des choix de représentation, qui correspondent à la définition du langage utilisé. Du point de vue sémantique, on travaille indépendamment du choix de représentation, "à un codage près".

L'intérêt des catégories réside dans le fait que l'on travaille le plus souvent "à un isomorphisme près", ce qui correspond à travailler "à un codage près" en informatique. Par exemple, en théorie des catégories, on définit les colimites à un isomorphisme près : un diagramme peut avoir plusieurs colimites, qui sont alors isomorphes. Lorsque l'on parle de "la" colimite d'un diagramme, il s'agit en fait souvent d'"une" colimite quelconque d'un diagramme, sachant que tant que l'on travaille à un isomorphisme près, on peut en choisir une quelconque. L'ensemble $(\{1\} \times A) \cup (\{2\} \times B)$ est un exemple de choix de colimite pour $A + B$ dans la catégorie des ensembles.

Ce degré de liberté est trop important lorsque l'on parle de syntaxe. On a en effet besoin de faire des choix de représentation ou de codage. Si on veut décrire une syntaxe pour des constructions de colimites, on ne peut plus parler des colimites à un isomorphisme près. En effet le fait d'avoir une représentation syntaxique par exemple pour une somme impose que l'on ait fait un choix particulier de somme.

Cela explique pourquoi dans toute la suite, nous parlons d'"objet initial choisi", ou de "somme amalgamée choisie". Techniquement, on doit faire des choix afin d'avoir une construction *libre*. L'introduction de colimites *choisies* dans une catégorie, due à C. Ehresmann [Ehr65, Ehr68], permet de donner un statut algébrique à cette catégorie.

3.1 Précatégories, préfoncteurs et pré-colimites

Nous commençons par définir la notion de précatégorie, similaire à celle de catégorie. Dans une précatégorie, il n'y a, *a priori*, pas d'égalité entre flèches, mais seulement des *équivalences*. Chaque ensemble de flèches est donc muni d'une relation de d'équivalence.

Définition 3.1 (Précatégorie) Une *précatégorie* $\mathcal{C}$ est définie par

- une classe d'*objets* $\mathrm{Obj}(\mathcal{C})$, (si A est un objet de $\mathrm{Obj}(\mathcal{C})$, on notera $A \in \mathrm{Obj}(\mathcal{C})$, même si $\mathrm{Obj}(\mathcal{C})$ n'est pas un ensemble) ;
- une famille $\mathrm{Arr}(\mathcal{C})$ indicée par $\mathrm{Obj}(\mathcal{C}) \times \mathrm{Obj}(\mathcal{C})$ de *flèches* — autrement dit, $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, on a un ensemble de flèches $\mathrm{Arr}(\mathcal{C})(A, B)$;
- une famille $\mathcal{R}(\mathcal{C})$ indicée par $\mathrm{Obj}(\mathcal{C}) \times \mathrm{Obj}(\mathcal{C})$ de relations sur $\mathrm{Arr}(\mathcal{C})$ — autrement dit, $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, on a une relation $\mathcal{R}(\mathcal{C})(A, B)$ sur l'ensemble $\mathrm{Arr}(\mathcal{C})(A, B)$;
- $\forall A, B, C \in \mathrm{Obj}(\mathcal{C})$, on a une opération de composition

$$\circ : \mathrm{Arr}(\mathcal{C})(B, C) \times \mathrm{Arr}(\mathcal{C})(A, B) \to \mathrm{Arr}(\mathcal{C})(A, C) ;$$

- $\forall A \in \mathrm{Obj}(\mathcal{C})$, on a une flèche identité $id_A \in \mathrm{Arr}(\mathcal{C})(A, A)$;
- $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, $\forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$,

$$\begin{array}{l}(f \circ id_A, f) \in \mathcal{R}(\mathcal{C})(A, B) \\ (id_B \circ f, f) \in \mathcal{R}(\mathcal{C})(A, B) ;\end{array}$$

- $\forall A, B, C, D \in \mathrm{Obj}(\mathcal{C})$,
  $\forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$, $\forall g \in \mathrm{Arr}(\mathcal{C})(B, C)$, $\forall h \in \mathrm{Arr}(\mathcal{C})(C, D)$,

$$((h \circ g) \circ f, h \circ (g \circ f)) \in \mathcal{R}(\mathcal{C})(A, D) ;$$

- $\mathcal{R}(\mathcal{C})$ est une congruence, c'est-à-dire
  - $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, $\mathcal{R}(\mathcal{C})(A, B)$ est une relation d'équivalence ;
  - $\forall A, B, C \in \mathrm{Obj}(\mathcal{C})$, $\forall f, f' \in \mathrm{Arr}(\mathcal{C})(A, B)$, $\forall g, g' \in \mathrm{Arr}(\mathcal{C})(B, C)$,

$$\left.\begin{array}{l}(f, f') \in \mathcal{R}(\mathcal{C})(A, B) \\ (g, g') \in \mathcal{R}(\mathcal{C})(B, C)\end{array}\right\} \Rightarrow (g \circ f, g' \circ f') \in \mathcal{R}(\mathcal{C})(A, C).$$

Lemme 3.1 *On peut associer à toute précatégorie $\mathcal{C}$ une* catégorie, *qu'on notera également, par abus de langage, $\mathcal{C}$. Cette catégorie est définie de la façon suivante.*

- *Les objets de la catégorie $\mathcal{C}$ sont les objets de la précatégorie $\mathcal{C}$.*
- *Comme $\mathcal{R}(\mathcal{C})(A, B)$ est une relation d'équivalence sur* $\mathrm{Arr}(\mathcal{C})(A, B)$, *on peut définir l'ensemble des flèches de A vers B de la catégorie $\mathcal{C}$ comme l'ensemble quotient*

  $$hom_{\mathcal{C}}(A, B) = \mathrm{Arr}(\mathcal{C})(A, B)/\mathcal{R}(\mathcal{C})(A, B).$$

  On notera $[f] : A \to B$ la classe d'équivalence de $f \in \mathrm{Arr}(\mathcal{C})(A, B)$.
- *Étant donné un objet A de $\mathcal{C}$, l'identité sur A est $[id_A] : A \to A$.*
- *La composition de deux flèches $[f] : A \to B$ et $[g] : B \to C$ est la classe $[g \circ f] : A \to C$. Cette définition est indépendante des représentants f et g choisis car $\mathcal{R}(\mathcal{C})$ est une congruence.*

Preuve. On vérifie immédiatement que

- les flèches $[id_A] : A \to A$ sont bien des identités;
- la composition est associative.

□

Réciproquement, on peut considérer toute catégorie comme une précatégorie, en prenant pour relation d'équivalence la relation d'égalité sur les flèches.

Définition 3.2 (Identité syntaxique) Soit une précatégorie $\mathcal{C}$. L'*égalité* dans $\mathcal{C}$ (à ne pas confondre avec l'équivalence $\mathcal{R}(\mathcal{C})$) est appelée *identité syntaxique.*

En effet, dans une précatégorie de termes, deux termes sont égaux lorsqu'ils sont *syntaxiquement identiques*, c'est-à-dire composés des mêmes symboles.

Notations

- L'identité syntaxique sera notée $\equiv$.
- Pour plus de lisibilité, et comme deux flèches équivalentes dans la précatégorie $\mathcal{C}$ sont *égales* dans la catégorie $\mathcal{C}$ associée, $(f, f') \in \mathcal{R}(\mathcal{C})(A, B)$ sera noté

  $$f = f' \in \mathrm{Arr}(\mathcal{C})(A, B).$$

  Nous rappelons le plus souvent possible l'ensemble auquel les flèches appartiennent, parce que nous manipulons plusieurs précatégories, et certaines flèches appartiennent à plusieurs ensembles de flèches dans des précatégories différentes.

Définition 3.3 (Petite précatégorie) Une précatégorie $\mathcal{C}$ est *petite* si et seulement si la catégorie associée est petite, c'est-à-dire si et seulement si $\mathrm{Obj}(\mathcal{C})$ est un ensemble.

L'analogue d'un foncteur entre deux catégories est un *préfoncteur* entre deux précatégories.

Définition 3.4 (Préfoncteur) Soit deux précatégories $\mathcal{C}$ et $\mathcal{D}$. Un *préfoncteur* F de $\mathcal{C}$ vers $\mathcal{D}$, noté $F : \mathcal{C} \to \mathcal{D}$ est une application qui à tout objet A de $\mathcal{C}$ associe un objet $F(A)$ de $\mathcal{D}$, et à toute flèche $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ associe une flèche $F(f) \in \mathrm{Arr}(\mathcal{D})(F(A), F(B))$, et telle que

- $\forall f, f' \in \mathrm{Arr}(\mathcal{C})(A, B)$,

$$f = f' \in \mathrm{Arr}(\mathcal{C})(A, B) \;\Rightarrow\; F(f) = F(f') \in \mathrm{Arr}(\mathcal{D})(F(A), F(B))\,;$$

- $\forall A \in \mathrm{Obj}(\mathcal{C})$, $F(id_A) = id_{F(A)} \in \mathrm{Arr}(\mathcal{D})(F(A), F(A))$;

- $\forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$ et $\forall g \in \mathrm{Arr}(\mathcal{C})(B, C)$,

$$F(g \circ f) = F(g) \circ F(f) \in \mathrm{Arr}(\mathcal{D})(F(A), F(C)).$$

À tout préfoncteur entre deux précatégories on peut associer un *foncteur* entre les catégories correspondantes.

Lemme 3.2 *Soit deux précatégories $\mathcal{C}$ et $\mathcal{D}$ et un préfoncteur $F : \mathcal{C} \to \mathcal{D}$. Alors on a un* foncteur, *noté également $F : \mathcal{C} \to \mathcal{D}$ entre les* catégories $\mathcal{C}$ *et $\mathcal{D}$, défini par*

$$\begin{array}{rccc} F & : & \mathcal{C} & \to & \mathcal{D} \\ & & A & \mapsto & A \\ & & [f] & \mapsto & [F(f)] \end{array}$$

La définition de $F([f])$ est bien indépendante du représentant f choisi car

$$f = f' \in \mathrm{Arr}(\mathcal{C})(A, B) \;\Rightarrow\; F(f) = F(f') \in \mathrm{Arr}(\mathcal{D})(F(A), F(B)).$$

Deux préfoncteurs $F, G : \mathcal{C} \to \mathcal{D}$ sont *syntaxiquement identiques* lorsque

- $\forall A \in \mathrm{Obj}(\mathcal{C})$, $F(A) \equiv G(A)$;

- $\forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$, $F(f) \equiv G(f)$.

On note $F \equiv G$.

On peut reformuler de nombreuses notions de théorie des catégories en les adaptant aux précatégories. Il suffit de remplacer les égalités entre flèches dans chaque catégorie par des équivalences dans la précatégorie correspondante.

Définition 3.5 (Isomorphisme) Soit une précatégorie $\mathcal{C}$. La flèche $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ est un *isomorphisme* (dans la précatégorie $\mathcal{C}$) si et seulement si il existe une flèche $g \in \mathrm{Arr}(\mathcal{C})(B, A)$ telle que $g \circ f = id_A \in \mathrm{Arr}(\mathcal{C})(A, A)$ et $f \circ g = id_B \in \mathrm{Arr}(\mathcal{C})(B, B)$.

On vérifie facilement qu'une flèche f dans une précatégorie $\mathcal{C}$ est un isomorphisme si et seulement si $[f]$ est un isomorphisme dans la catégorie associée.

Définition 3.6 (Préfoncteur plein) Soit deux précatégories $\mathcal{C}$ et $\mathcal{D}$, et un préfoncteur $F : \mathcal{C} \to \mathcal{D}$. Le préfoncteur F est *plein* si et seulement si pour toute flèche $g \in \mathrm{Arr}(\mathcal{D})(F(A), F(B))$. il existe une flèche $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ telle que

$$g = F(f) \in \mathrm{Arr}(\mathcal{D})(F(A), F(B)).$$

Définition 3.7 (Préfoncteur fidèle) Soit deux précatégories $\mathcal{C}$ et $\mathcal{D}$, et un préfoncteur $F : \mathcal{C} \to \mathcal{D}$. Le préfoncteur F est *fidèle* si et seulement si pour toutes flèches $f, f' \in \mathrm{Arr}(\mathcal{C})(A, B)$,

$$F(f) = F(f') \in \mathrm{Arr}(\mathcal{D})(F(A), F(B)) \Rightarrow f = f' \in \mathrm{Arr}(\mathcal{C})(A, B).$$

On vérifie facilement qu'un préfoncteur F est *plein* (respectivement *fidèle*) si et seulement si le foncteur associé est *plein* (respectivement *fidèle*).

On peut enfin reformuler la notion de colimite d'un diagramme pour une précatégorie. Nous définissons ici uniquement les notions d'*objet pré-initial* et de *pré-sommes amalgamées.*

Définition 3.8 (Objet pré-initial) Soit une précatégorie $\mathcal{C}$. Un objet $\emptyset$ de $\mathcal{C}$ est *pré-initial* si et seulement si

1. $\forall A \in \mathrm{Obj}(\mathcal{C})$, il existe une flèche $j_A \in \mathrm{Arr}(\mathcal{C})(\emptyset, A)$;
2. $\forall f \in \mathrm{Arr}(\mathcal{C})(\emptyset, A)$, $f = j_A \in \mathrm{Arr}(\mathcal{C})(\emptyset, A)$.

Lemme 3.3 *Un objet $\emptyset$ est pré-initial dans une précatégorie $\mathcal{C}$ si et seulement si $\emptyset$ est initial dans la catégorie associée $\mathcal{C}$.*

Définition 3.9 (Pré-somme amalgamée) Soit une précatégorie $\mathcal{C}$. Soit trois objets A, B et C de $\mathcal{C}$, ainsi que deux flèches $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ et $g \in \mathrm{Arr}(\mathcal{C})(A, C)$. Le triplet $(P, \&_1, \&_2)$, où $P \in \mathrm{Obj}(\mathcal{C})$, $\&_1 \in \mathrm{Arr}(\mathcal{C})(B, P)$ et $\&_2 \in \mathrm{Arr}(\mathcal{C})(C, P)$, est une *pré-somme amalgamée* de B et C par rapport aux flèches f et g si et seulement si

1. $\&_1 \circ f = \&_2 \circ g \in \mathrm{Arr}(\mathcal{C})(A, P)$;
2. si $D \in \mathrm{Obj}(\mathcal{C})$, $f' \in \mathrm{Arr}(\mathcal{C})(B, D)$, $g' \in \mathrm{Arr}(\mathcal{C})(C, D)$ et $f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C})(A, D)$, alors,
   (a) il existe une flèche $u \in \mathrm{Arr}(\mathcal{C})(P, D)$ telle que
   $$u \circ \&_1 = f' \in \mathrm{Arr}(\mathcal{C})(B, D)$$
   $$u \circ \&_2 = g' \in \mathrm{Arr}(\mathcal{C})(C, D) ;$$
   (b) pour toute flèche $v \in \mathrm{Arr}(\mathcal{C})(P, D)$ telle que
   $$v \circ \&_1 = f' \in \mathrm{Arr}(\mathcal{C})(B, D)$$
   $$v \circ \&_2 = g' \in \mathrm{Arr}(\mathcal{C})(C, D),$$
   on a $u = v \in \mathrm{Arr}(\mathcal{C})(P, D)$.

Lemme 3.4 *Le triplet* $(P, \&_1, \&_2)$ *est une pré-somme amalgamée de* B *et* C *par rapport aux flèches* f *et* g *dans la précatégorie* $\mathcal{C}$ *si et seulement si le triplet* $(P, [\&_1], [\&_2])$ *est une somme amalgamée de* B *et* C *par rapport aux flèches* $[f]$ *et* $[g]$ *dans la catégorie associée* $\mathcal{C}$.

On peut définir, de façon similaire, la *pré-colimite* d'un diagramme sur une précatégorie. Une précatégorie est *finiment pré-cocomplète* lorsque tout diagramme fini a une pré-colimite.

De façon générale, une pré-colimite dans une précatégorie correspond à une colimite dans la catégorie associée. Par conséquent, une précatégorie est finiment pré-cocomplète si et seulement si la catégorie associée est finiment cocomplète. De plus, une précatégorie est finiment pré-cocomplète si et seulement si elle a un objet pré-initial et des pré-sommes amalgamées.

Précatégorie DIAGR($\mathcal{C}$)

Étant donné une catégorie $\mathcal{C}$, la catégorie DIAGR($\mathcal{C}$), avec la relation de congruence $\approx$, est une précatégorie. Plus précisément,

- l'identité syntaxique dans la précatégorie DIAGR($\mathcal{C}$) est l'égalité dans la catégorie DIAGR($\mathcal{C}$) :
  - si $\overline{\alpha}$ et $\overline{\beta}$ sont deux objets de DIAGR($\mathcal{C}$),
    $$\overline{\alpha} \equiv \overline{\beta} \text{ (dans la précatégorie)} \Leftrightarrow \overline{\alpha} = \overline{\beta} \text{ (dans la catégorie)} ;$$
  - si $\overline{\sigma}, \overline{\tau} : \overline{\alpha} \to \overline{\beta}$ sont deux flèches de DIAGR($\mathcal{C}_0$),
    $$\overline{\sigma} \equiv \overline{\tau} \text{ (dans la précatégorie)} \Leftrightarrow \overline{\sigma} = \overline{\tau} \text{ (dans la catégorie)} ;$$
- la relation d'équivalence dans la précatégorie DIAGR($\mathcal{C}$) est la congruence $\approx$ sur les flèches de la catégorie DIAGR($\mathcal{C}$) : si $\overline{\sigma}, \overline{\tau} : \overline{\alpha} \to \overline{\beta}$ sont deux flèches de DIAGR($\mathcal{C}_0$),
  $$\overline{\sigma} = \overline{\tau} \in \mathrm{Arr}(\mathrm{DIAGR}(\mathcal{C}))(\overline{\alpha}, \overline{\beta}) \Leftrightarrow \overline{\sigma} \approx \overline{\tau}.$$

La catégorie associée à la précatégorie DIAGR($\mathcal{C}$) est diagr($\mathcal{C}$). De plus, le foncteur $I_{\mathcal{C}} : \mathcal{C} \to \mathrm{DIAGR}(\mathcal{C})$ peut être considéré comme un préfoncteur dont le foncteur associé est $[-]_{\mathcal{C}} \circ I_{\mathcal{C}} : \mathcal{C} \to \mathrm{diagr}(\mathcal{C})$.

D'autre part, la précatégorie DIAGR($\mathcal{C}$) est finiment pré-cocomplète, d'après les lemmes 2.17 (partie I, page 100) et 2.19 (partie I, page 101). Remarquons que ce sont justement ces deux lemmes qui nous ont permis de montrer que la catégorie diagr($\mathcal{C}$) est finiment cocomplète. L'aplatissement

$$\mathrm{Apl}_{\mathcal{C}} : \mathrm{DIAGR}^2(\mathcal{C}) \to \mathrm{DIAGR}(\mathcal{C})$$

donne un choix de pré-colimites dans DIAGR($\mathcal{C}$). En particulier, la précatégorie DIAGR($\mathcal{C}$) a un objet pré-initial choisi (le diagramme vide) et des pré-sommes amalgamées choisies (obtenues par aplatissement).

3.2 Problème de circularité termes – congruences

Soit $\mathcal{C}_0$ une petite catégorie. Nous cherchons à définir une catégorie qui contient $\mathcal{C}_0$, qui a un objet initial et des sommes amalgamées. Nous allons en fait définir une précatégorie avec un objet pré-initial et des pré-sommes amalgamées.

Objet pré-initial

Pour l'objet pré-initial, il nous suffit d'introduire un nouvel objet $\emptyset$, et pour tout objet A, une flèche j_A de $\emptyset$ vers A. De plus, pour chaque couple de flèches (f, g) de $\emptyset$ vers A, on introduit la relation $f = g$.

Pré-sommes amalgamées

Étant donné trois objets A, B et C, ainsi que deux flèches $f : A \to B$ et $g : A \to C$, nous introduisons un nouvel objet

$$\mathsf{push}\,(A, B, C, f, g)$$

deux flèches

$$\&_1(A, B, C, f, g) : B \to \mathsf{push}\,(A, B, C, f, g)$$
$$\&_2(A, B, C, f, g) : C \to \mathsf{push}\,(A, B, C, f, g)$$

et la relation

$$\&_1(A, B, C, f, g) \circ f = \&_2(A, B, C, f, g) \circ g.$$

Il y a donc une circularité entre la définition des objets et la définition des flèches, puisque l'introduction d'une nouvelle flèche permet d'introduire ensuite des nouveaux objets. Cette circularité ne pose, *a priori*, pas de problème car il s'agit d'une circularité au niveau des *termes*.
De plus, étant donné un objet D et deux flèches $f' : B \to D$ et $g' : C \to D$ telles que

$$f' \circ f = g' \circ g$$

on introduit une flèche

$$\mathsf{up}\,(A, B, C, D, f, g, f', g') : \mathsf{push}\,(A, B, C, f, g) \to D$$

et deux relations

$$\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g) = f'$$
$$\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_2(A, B, C, f, g) = g'.$$

Il faut également spécifier qu'il y a une unique flèche qui satisfait ces deux relations. L'introduction de la flèche

$$\mathsf{up}\,(A, B, C, D, f, g, f', g') : \mathsf{push}\,(A, B, C, f, g) \to D$$

pose un problème. Jusqu'à présent, nous avons défini d'abord des termes, puis des relations entre les termes. Ici, nous devons introduire un nouveau terme *à condition*

qu'une relation soit vérifiée. Il y a donc une circularité entre la définition des termes et la définition des relations.

Les théories algébriques généralisées, qui sont une généralisation des algèbres multi-sortes, ont été proposées par Cartmell pour spécifier des types *dépendants*, c'est-à-dire paramétrés par des termes [Car86]. Par exemple le "type" $\mathrm{Arr}(A, B)$ dépend de deux termes A et B. Cependant, les types dépendants de Cartmell ne permettent pas de spécifier des termes dont l'existence dépend d'une équivalence entre deux autres termes. T. Streicher et M. Wirsing, qui préconisent l'utilisation de types dépendants [SW91], ne précisent pas comment résoudre ce problème.

Une solution est de remplacer la flèche **up** qui pose problème par des flèches dont l'existence ne dépend pas d'une équivalence entre deux termes. Cette approche a été proposée par F. Cury [Cur91]. Les flèches **up** sont remplacées par deux flèches **p** et **d**, qui sont des flèches **up** particulières dont l'existence n'est pas conditionnée par une équivalence. Toute flèche **up** peut être reconstruite *a posteriori* à l'aide des flèches **p** et **d**.

Dans notre travail, nous voulons d'une part rester proche de la définition classique de la somme amalgamée, et d'autre part ne pas multiplier les flèches et les règles à considérer. Pour ces raisons, nous conservons les flèches **up** et proposons une construction stratifiée de la syntaxe, qui permet d'éliminer le problème de circularité. Nous définissons une suite de précatégories $\mathcal{C}_i$ telles que, dans la précatégorie $\mathcal{C}_{i+1}$, l'introduction d'une flèche **up** dépend uniquement d'une relation dans $\mathcal{C}_i$. Étant donné des objets A, B, C, D, et des flèches $f : A \to B$, $g : A \to C$, $f' : B \to D$ et $g' : C \to D$ de $\mathcal{C}_i$, tels qu'on ait la relation $f \circ f' = g \circ g'$ dans $\mathcal{C}_i$, on introduit une flèche

$$\mathbf{up}\,(A, B, C, D, f, g, f', g') : \mathbf{push}\,(A, B, C, f, g) \to D$$

dans $\mathcal{C}_{i+1}$. Dans chaque $\mathcal{C}_i$, nous définissons donc d'abord les objets — l'ensemble $\mathrm{Obj}(\mathcal{C}_i)$ —, ensuite les flèches — la famille d'ensembles $\mathrm{Arr}(\mathcal{C}_i)$ —, et enfin les équivalences entre flèches — la famille de relations $\mathcal{R}(\mathcal{C}_i)$.

3.3 Précatégories $\mathcal{C}_i$

Soit $\mathcal{C}_0$ une petite catégorie. On considère la précatégorie associée $\mathcal{C}_0$. On a donc un ensemble d'objets $\mathrm{Obj}(\mathcal{C}_0)$, une famille de flèches $\mathrm{Arr}(\mathcal{C}_0)$ et une famille de relations $\mathcal{R}(\mathcal{C}_0)$. Si A et B sont des objets de $\mathcal{C}_0$, la relation $\mathcal{R}(\mathcal{C}_0)(A, B)$ sur l'ensemble $\mathrm{Arr}(\mathcal{C}_0)(A, B)$ est l'égalité dans la catégorie $\mathcal{C}_0$.

Dans ce paragraphe, nous définissons, pour tout entier naturel i, une petite précatégorie $\mathcal{C}_{i+1}$, en définissant successivement les ensembles et familles d'ensembles $\mathrm{Obj}(\mathcal{C}_{i+1})$, $\mathrm{Arr}(\mathcal{C}_{i+1})$ et $\mathcal{R}(\mathcal{C}_{i+1})$, à partir de $\mathrm{Obj}(\mathcal{C}_i)$, $\mathrm{Arr}(\mathcal{C}_i)$ et $\mathcal{R}(\mathcal{C}_i)$. Chaque ensemble est caractérisé par des règles de la forme

$$\frac{x \in X \;;\; y \in Y}{\mathbf{z}(x, y) \in Z}$$

Cette règle définit partiellement l'ensemble Z, et signifie :

> "Si le terme x appartient à X, et si le terme y appartient à Y, alors le terme $\mathsf{z}(x, y)$ appartient à Z."

Ensuite, on considère le plus petit ensemble Z qui satisfait toutes les règles qui le définissent.

Règles qui définissent l'ensemble $\mathrm{Obj}(\mathcal{C}_{i+1})$

Les objets de $\mathcal{C}_{i+1}$ sont les objets de $\mathcal{C}_0$ (règle (1)), et les objets que l'on peut construire à partir d'objets et de flèches de $\mathcal{C}_i$, c'est-à-dire objet pré-initial (règle (2)) et pré-somme amalgamée (règle (3)).

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_0)}{A \in \mathrm{Obj}(\mathcal{C}_{i+1})} \tag{1}$$

$$\frac{}{\emptyset \in \mathrm{Obj}(\mathcal{C}_{i+1})} \tag{2}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A, C)}{\mathsf{push}\,(A, B, C, f, g) \in \mathrm{Obj}(\mathcal{C}_{i+1})} \tag{3}$$

Règles qui définissent la famille d'ensembles $\mathrm{Arr}(\mathcal{C}_{i+1})$

Les flèches de $\mathcal{C}_{i+1}$ sont les flèches de $\mathcal{C}_0$ (règle (4)), les compositions de flèches de $\mathcal{C}_{i+1}$ (règle (5)), les identités entre objets de $\mathcal{C}_{i+1}$ (règle (6)), les flèches de cône colimite (règles (8), (9)), et les uniques flèches d'un objet colimite vers le sommet d'un cône (règles (7), (10)).

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_0)(A, B)}{f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{4}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, C)}{g \circ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, C)} \tag{5}$$

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_{i+1})}{\mathsf{id}_A \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, A)} \tag{6}$$

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_{i+1})}{\mathsf{j}_A \in \mathrm{Arr}(\mathcal{C}_{i+1})(\emptyset, A)} \tag{7}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A, C)}{\&_1(A, B, C, f, g) \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, \mathsf{push}\,(A, B, C, f, g))} \tag{8}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A, C)}{\&_2(A, B, C, f, g) \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, \mathsf{push}\,(A, B, C, f, g))} \tag{9}$$

$$\frac{\begin{array}{c}A, B, C, D \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A, C) \\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B, D) \ ; \ g' \in \mathrm{Arr}(\mathcal{C}_i)(C, D) \ ; \ f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A, D)\end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') \in \mathrm{Arr}(\mathcal{C}_{i+1})(\mathsf{push}\,(A, B, C, f, g), D)} \tag{10}$$

Remarquons que l'existence de la flèche up de $\mathcal{C}_{i+1}$ dépend d'une relation entre deux flèches de $\mathcal{C}_i$. L'introduction d'un terme dans une précatégorie ne dépend jamais d'une relation dans cette précatégorie.

Règles qui définissent la famille de relations $\mathcal{R}(\mathcal{C}_{i+1})$

La relation $\mathcal{R}(\mathcal{C}_{i+1})$ contient la relation $\mathcal{R}(\mathcal{C}_0)$ (règle (11)). $\mathcal{R}(\mathcal{C}_{i+1})$ est une relation d'équivalence (règles (12)-(14)), et une congruence (règle (15)). Pour que $\mathcal{C}_{i+1}$ soit une précatégorie, il faut que la composition soit associative (règle (16)), et que les flèches id_A soient des flèches identités (règles (17), (18)). Les règles (19)-(23) servent à assurer que $\emptyset$ est pré-initial, et que push (A, B, C, f, g) est une pré-somme amalgamée.

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ f, g \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \quad f = g \in \mathrm{Arr}(\mathcal{C}_0)(A, B)}{f = g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{11}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{f = f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{12}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f, g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \quad f = g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{g = f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{13}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f, g, h \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \quad f = g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \quad g = h \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{f = h \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{14}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f, f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \ ; \ g, g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, C) \quad f = f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \ ; \ g = g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, C)}{g \circ f = g' \circ f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, C)} \tag{15}$$

$$\frac{A, B, C, D \in \mathrm{Obj}(\mathcal{C}_{i+1}) \quad f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, C) \ ; \ h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, D)}{(h \circ g) \circ f = h \circ (g \circ f) \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, D)} \tag{16}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{f \circ \mathsf{id}_A = f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{17}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{\mathsf{id}_B \circ f = f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)} \tag{18}$$

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f, g \in \mathrm{Arr}(\mathcal{C}_{i+1})(\emptyset, A)}{f = g \in \mathrm{Arr}(\mathcal{C}_{i+1})(\emptyset, A)} \tag{19}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A, C)}{\&_1(A, B, C, f, g) \circ f = \&_2(A, B, C, f, g) \circ g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, \mathsf{push}\,(A, B, C, f, g))} \tag{20}$$

$$\frac{\begin{array}{c}A,B,C,D \in \mathrm{Obj}(\mathcal{C}_i)\ ;\ f \in \mathrm{Arr}(\mathcal{C}_i)(A,B)\ ;\ g \in \mathrm{Arr}(\mathcal{C}_i)(A,C)\\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B,D)\ ;\ g' \in \mathrm{Arr}(\mathcal{C}_i)(C,D)\ ;\ f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A,D)\end{array}}{\mathsf{up}\,(A,B,C,D,f,g,f',g') \circ \&_1(A,B,C,f,g) = f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B,D)} \quad (21)$$

$$\frac{\begin{array}{c}A,B,C,D \in \mathrm{Obj}(\mathcal{C}_i)\ ;\ f \in \mathrm{Arr}(\mathcal{C}_i)(A,B)\ ;\ g \in \mathrm{Arr}(\mathcal{C}_i)(A,C)\\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B,D)\ ;\ g' \in \mathrm{Arr}(\mathcal{C}_i)(C,D)\ ;\ f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A,D)\end{array}}{\mathsf{up}\,(A,B,C,D,f,g,f',g') \circ \&_2(A,B,C,f,g) = g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(C,D)} \quad (22)$$

$$\frac{\begin{array}{c}A,B,C,D \in \mathrm{Obj}(\mathcal{C}_i)\ ;\ f \in \mathrm{Arr}(\mathcal{C}_i)(A,B)\ ;\ g \in \mathrm{Arr}(\mathcal{C}_i)(A,C)\\ u,v \in \mathrm{Arr}(\mathcal{C}_{i+1})(\mathsf{push}\,(A,B,C,f,g),D)\\ u \circ \&_1(A,B,C,f,g) = v \circ \&_1(A,B,C,f,g) \in \mathrm{Arr}(\mathcal{C}_{i+1})(B,D)\\ u \circ \&_2(A,B,C,f,g) = v \circ \&_2(A,B,C,f,g) \in \mathrm{Arr}(\mathcal{C}_{i+1})(C,D)\end{array}}{u = v \in \mathrm{Arr}(\mathcal{C}_{i+1})(\mathsf{push}\,(A,B,C,f,g),D)} \quad (23)$$

Pour obtenir une suite de précatégories $\mathcal{C}_i$, il faut préciser les flèches identités et les compositions. Les flèches identités sont évidemment les termes $\mathsf{id}_A \in \mathrm{Arr}(\mathcal{C}_i)(A,A)$ et les opérations de composition sont les fonctions

$$\begin{array}{rccc} \circ\ : & \mathrm{Arr}(\mathcal{C}_i)(B,C) \times \mathrm{Arr}(\mathcal{C}_i)(A,B) & \to & \mathrm{Arr}(\mathcal{C}_i)(A,C) \\ & (g,f) & \mapsto & g \circ f. \end{array}$$

Lemme 3.5 *Pour tout $i \geq 0$, on a:*

1. $\mathrm{Obj}(\mathcal{C}_i) \subseteq \mathrm{Obj}(\mathcal{C}_{i+1})$ *:*
2. $\forall A,B \in \mathrm{Obj}(\mathcal{C}_i).\ \mathrm{Arr}(\mathcal{C}_i)(A,B) \subseteq \mathrm{Arr}(\mathcal{C}_{i+1})(A,B)$ *;*
3. $\forall A,B \in \mathrm{Obj}(\mathcal{C}_i),\ \mathcal{R}(\mathcal{C}_i)(A,B) \subseteq \mathcal{R}(\mathcal{C}_{i+1})(A,B)$.

Preuve. Par induction sur i. Pour $i = 0$, le point 1 provient de la règle (1), le point 2 provient de la règle (4), et le point 3 provient de la règle (11). Pour $i+1$ on prouve les trois points en parallèle, par induction sur la structure des éléments de $\mathrm{Obj}(\mathcal{C}_{i+1})$, $\mathrm{Arr}(\mathcal{C}_{i+1})$ et $\mathcal{R}(\mathcal{C}_{i+1})$. Cela nécessite de considérer toutes les règles (1)–(23). □

Lemme 3.6 *Pour tout $i \geq 0$, on a:*

1. $\mathcal{R}(\mathcal{C}_{i+1})$ *est une congruence;*
2. $\mathcal{C}_{i+1}$ *est une précatégorie;*
3. *l'objet $\emptyset$ est pré-initial dans $\mathcal{C}_{i+1}$.*

Preuve. Toutes les propositions dérivent de façon évidente des règles qui définissent $\mathrm{Obj}(\mathcal{C}_{i+1})$, $\mathrm{Arr}(\mathcal{C}_{i+1})$ et $\mathcal{R}(\mathcal{C}_{i+1})$. Plus précisément :

1. règles (12)–(15) ;
2. règles (5), (6) et (16)–(18) ;
3. règles (7) et (19).

□

Exemple 3.1 Supposons que $\mathcal{C}_0$ contient trois objets A, B et C, et deux flèches $f : A \to B$ et $g : A \to C$. On peut construire une pré-somme amalgamée de deux façons différentes :

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_0)(A, C)}{P_1 \equiv \mathsf{push}\,(A, B, C, f, g) \in \mathrm{Obj}(\mathcal{C}_1)} \quad (3)$$

$$\frac{A, C, B, \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_0)(A, C) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_0)(A, B)}{P_2 \equiv \mathsf{push}\,(A, C, B, g, f) \in \mathrm{Obj}(\mathcal{C}_1)} \quad (3)$$

On a également :

$$\frac{A, C, B, \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_0)(A, C) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_0)(A, B)}{\&_1(A, C, B, g, f) \circ g = \&_2(A, C, B, g, f) \circ f \in \mathrm{Arr}(\mathcal{C}_1)(A, P_2)} \quad (20)$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_0) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_0)(A, C)}{\&_1(A, B, C, f, g) \circ f = \&_2(A, B, C, f, g) \circ g \in \mathrm{Arr}(\mathcal{C}_1)(A, P_1)} \quad (20)$$

Par la règle (10), on peut donc construire les termes

$$u \equiv \mathsf{up}\,(A, B, C, P_2, f, g, \&_1(A, C, B, g, f), \&_2(A, C, B, g, f)) \in \mathrm{Arr}(\mathcal{C}_2)(P_1, P_2),$$
$$v \equiv \mathsf{up}\,(A, C, B, P_1, f, g, \&_1(A, B, C, f, g), \&_2(A, B, C, f, g)) \in \mathrm{Arr}(\mathcal{C}_2)(P_2, P_1).$$

D'après la règle (12), on a

$$\&_1(A, B, C, f, g) = \&_1(A, B, C, f, g) \in \mathrm{Arr}(\mathcal{C}_1)(B, P_1),$$
$$\&_2(A, B, C, f, g) = \&_2(A, B, C, f, g) \in \mathrm{Arr}(\mathcal{C}_1)(C, P_1).$$

En appliquant la règle (23), on en déduit que

$$u \circ v = \mathsf{id}_{P_2} \in \mathrm{Arr}(\mathcal{C}_2)(P_2, P_2).$$

De même, on montre que

$$v \circ u = \mathsf{id}_{P_1} \in \mathrm{Arr}(\mathcal{C}_2)(P_1, P_1).$$

Nous en déduisons donc que les deux constructions P_1 et P_2 sont isomorphes dans la précatégorie $\mathcal{C}_2$, (mais P_1 et P_2 ne sont pas isomorphes dans la précatégorie $\mathcal{C}_1$).

Remarque 3.1 Les triplets

$$(\mathsf{push}\,(A,B,C,f,g),\ \&_1(A,B,C,f,g),\ \&_2(A,B,C,f,g))$$

ne sont pas, en général, des pré-sommes amalgamées. Le terme $\mathsf{push}\,(A,B,C,f,g)$, dans l'exemple 3.1, n'est pas une pré-somme amalgamée dans $\mathcal{C}_1$ car la flèche u ne fait pas partie de $\mathcal{C}_1$. Par contre, lorsque nous considérerons la précatégorie des termes, limite de la suite de précatégories $\mathcal{C}_i$ (section 3.6), les triplets

$$(\mathsf{push}\,(A,B,C,f,g),\ \&_1(A,B,C,f,g),\ \&_2(A,B,C,f,g))$$

seront bien des pré-sommes amalgamées.

Lemme 3.7 *L'ensemble des flèches d'un objet de $\mathcal{C}_0$ vers l'objet pré-initial est vide. Autrement dit,*

$$\forall i \geq 1,\ \forall A \in \mathrm{Obj}(\mathcal{C}_0),\ \mathrm{Arr}(\mathcal{C}_i)(A,\emptyset) = \varnothing.$$

Preuve. Par induction sur la définition de $\mathrm{Arr}(\mathcal{C}_i)$. □

3.4 Simplification des flèches

Dans cette section, nous définissons pour tout entier $i \geq 0$, un *système de simplification* des flèches de la précatégorie $\mathcal{C}_{i+1}$. Il s'agit ici d'un outil technique qui nous permet de démontrer en section 3.5 que, pour tout i, la précatégorie $\mathcal{C}_i$ est une extension conservatrice de $\mathcal{C}_0$.

Définition 3.10 (Flèche élémentaire) On dit qu'une flèche $f_j \in \mathrm{Arr}(\mathcal{C}_i)(A,B)$ est *élémentaire* si et seulement si l'une des conditions suivantes est satisfaite.

1. $f_j \in \mathrm{Arr}(\mathcal{C}_0)(A,B)$;
2. $f_j \equiv \&_k(A,B,C,f,g)$;
3. $f_j \equiv \mathsf{up}\,(A,B,C,D,f,g,f',g')$;
4. $f_j \equiv \mathsf{id}_X$ et $f_j \notin \mathrm{Arr}(\mathcal{C}_0)(X,X)$;
5. $f_j \equiv \mathsf{j}_X$.

Dans tout le reste de cette section, nous considérons les flèches de $\mathcal{C}_i$ comme des compositions

$$f \equiv f_n \circ f_{n-1} \circ \cdots \circ f_1$$

avec, pour tout $j \in \{1,..n\}$, f_j est une flèche élémentaire. L'entier n est appelé *longueur* du terme f.

Cette présentation simplifiée est justifiée par la règle (16), c'est-à-dire l'associativité de la composition.

Le système de simplification est composé de *règles de simplification*, qui définissent une relation $\longrightarrow$ sur les flèches de la précatégorie $\mathcal{C}_{i+1}$. On considère ensuite la fermeture réflexive et transitive, notée $\longrightarrow^*$, de la relation $\longrightarrow$.

Nous considérons des règles de simplification de l'une des deux formes suivantes.

$$\frac{P}{f \longrightarrow f'} \qquad \text{(Axiome)}$$

$$\frac{P}{u \longrightarrow u' \;\Rightarrow\; f \longrightarrow f'} \qquad \text{(Règle de contexte)}$$

où f, f', u et u' sont des flèches de la précatégorie $\mathcal{C}_{i+1}$, et P est l'ensemble des conditions qui assurent que les termes f, f', u et u' sont bien formés.

Intuitivement, si $f \longrightarrow^* f'$ (c'est-à-dire si la flèche f peut se simplifier en la flèche f'), alors f et f' sont équivalentes dans la précatégorie $\mathcal{C}_{i+1}$. La réciproque n'est pas vraie. Autrement dit, le système de simplification est cohérent, mais pas complet, par rapport à l'équivalence dans $\mathcal{C}_{i+1}$.

On considère le système de simplification des flèches de $\mathcal{C}_{i+1}$ composé des règles suivantes.

Axiomes

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \;;\; f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{f \circ \mathsf{id}_A \longrightarrow f} \qquad (\rightarrow 17)$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_{i+1}) \;;\; f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)}{\mathsf{id}_B \circ f \longrightarrow f} \qquad (\rightarrow 18)$$

$$\frac{\begin{array}{c}A, B, C, D \in \mathrm{Obj}(\mathcal{C}_i) \;;\; f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \;;\; g \in \mathrm{Arr}(\mathcal{C}_i)(A, C) \\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B, D) \;;\; g' \in \mathrm{Arr}(\mathcal{C}_i)(C, D) \;;\; f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A, D)\end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g) \longrightarrow f'} \qquad (\rightarrow 20)$$

$$\frac{\begin{array}{c}A, B, C, D \in \mathrm{Obj}(\mathcal{C}_i) \;;\; f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \;;\; g \in \mathrm{Arr}(\mathcal{C}_i)(A, C) \\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B, D) \;;\; g' \in \mathrm{Arr}(\mathcal{C}_i)(C, D) \;;\; f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A, D)\end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_2(A, B, C, f, g) \longrightarrow g'} \qquad (\rightarrow 21)$$

Règles de contexte

$$\frac{A,B,C \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f,f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A,B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_{i+1})(B,C)}{f \longrightarrow f' \Rightarrow g \circ f \longrightarrow g \circ f'} \quad (\rightarrow 15\,\mathrm{i})$$

$$\frac{A,B,C \in \mathrm{Obj}(\mathcal{C}_{i+1}) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A,B) \ ; \ g,g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B,C)}{g \longrightarrow g' \Rightarrow g \circ f \longrightarrow g' \circ f} \quad (\rightarrow 15\,\mathrm{ii})$$

$$\frac{\begin{array}{c}A,B,C,D \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A,B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A,C) \\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B,D) \ ; \ g' \in \mathrm{Arr}(\mathcal{C}_i)(C,D) \ ; \ f'' \in \mathrm{Arr}(\mathcal{C}_i)(B,D) \\ f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A,D)\end{array}}{f' \longrightarrow f'' \Rightarrow \mathsf{up}\,(A,B,C,D,f,g,f',g') \longrightarrow \mathsf{up}\,(A,B,C,D,f,g,f'',g')} \quad (\rightarrow 23\,\mathrm{i})$$

$$\frac{\begin{array}{c}A,B,C,D \in \mathrm{Obj}(\mathcal{C}_i) \ ; \ f \in \mathrm{Arr}(\mathcal{C}_i)(A,B) \ ; \ g \in \mathrm{Arr}(\mathcal{C}_i)(A,C) \\ f' \in \mathrm{Arr}(\mathcal{C}_i)(B,D) \ ; \ g' \in \mathrm{Arr}(\mathcal{C}_i)(C,D) \ ; \ g'' \in \mathrm{Arr}(\mathcal{C}_i)(C,D) \\ f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A,D)\end{array}}{g' \longrightarrow g'' \Rightarrow \mathsf{up}\,(A,B,C,D,f,g,f',g') \longrightarrow \mathsf{up}\,(A,B,C,D,f,g,f',g'')} \quad (\rightarrow 23\,\mathrm{ii})$$

Remarque 3.2 Ce système de simplification ressemble à un système de réécriture (cf. par exemple [DJ90]). Il y a cependant deux différences avec les systèmes de réécriture classiques.

1. Nous avons besoin de prémisses pour assurer que les termes manipulés sont bien formés.
2. Nous n'autorisons pas toutes les règles de passage au contexte. Nous avons une règle de passage au contexte uniquement pour l'opérateur de composition (règles ($\rightarrow$ 15i) et ($\rightarrow$ 15ii)), et pour les deux derniers paramètres de l'opérateur up (règles ($\rightarrow$ 23i) et ($\rightarrow$ 23ii)).

Lemme 3.8 *Soit* $f \in \mathrm{Arr}(\mathcal{C}_i)(A,B)$. *alors,*

1. $f \longrightarrow f' \Rightarrow f' \in \mathrm{Arr}(\mathcal{C}_i)(A,B)$;
2. $f \longrightarrow f' \Rightarrow f = f' \in \mathrm{Arr}(\mathcal{C}_i)(A,B)$.

Preuve. Aucune difficulté. □

Lemme 3.9 *(Terminaison du système de simplification) Le système de simplification termine, c'est-à-dire qu'il n'existe pas de séquence infinie*

$$f_1 \longrightarrow f_2 \longrightarrow \cdots f_n \longrightarrow \cdots.$$

Preuve. Pour chaque axiome

$$\frac{P}{f \longrightarrow f'},$$

f' est un sous terme de f. □

Lemme 3.10 *(Confluence du système de simplification)*
Le système de simplification est confluent, c'est-à-dire que si $f \longrightarrow f_1$ *et* $f \longrightarrow f_2$, *alors il existe* f' *tel que* $f_1 \longrightarrow^* f'$ *et* $f_2 \longrightarrow^* f'$.

Preuve. Aucune difficulté. □

Définition 3.11 (Forme normale) Soit une flèche $f \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$. On dit que la flèche f est sous *forme normale*, ou *irréductible*, si il n'existe aucune simplification $f \longrightarrow f'$.

Proposition 3.1 *Toute flèche* $f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$ *a une forme normale.*

Preuve. C'est une conséquence immédiate de la terminaison et confluence du système de simplification. □

La forme normale de f sera notée $\mathcal{N}(f)$.

Théorème 3.1 *(Cohérence de la simplification) Soit* $f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *On a*

$$f = \mathcal{N}(f) \in \mathrm{Arr}(\mathcal{C}_i)(A, B).$$

Preuve. Par induction sur la longueur de la simplification $f \longrightarrow^* \mathcal{N}(f)$.
Pour une simplification de longueur 0, le résultat est évident.
Supposons $f \longrightarrow f' \longrightarrow^* \mathcal{N}(f')$. D'après le lemme 3.8, $f = f' \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. Ensuite, par hypothèse d'induction, on a $f' = \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$, d'où $f = \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. □

Le théorème suivant affirme que toute flèche pour laquelle le domaine et le codomaine sont dans $\mathcal{C}_0$ a pour forme normale une flèche de $\mathcal{C}_0$.

Théorème 3.2 *(Forme normale (1))*
Soit $A, B \in \mathrm{Obj}(\mathcal{C}_0)$ *et* $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *On a*

$$\mathcal{N}(m) \in \mathrm{Arr}(\mathcal{C}_0)(A, B).$$

Preuve. Soit $A, B \in \mathrm{Obj}(\mathcal{C}_0)$ et $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. Soit

$$\mathcal{N}(m) \equiv m_n \circ m_{n-1} \cdots m_2 \circ m_1,$$

avec pour tout $j \in \{1, \ldots n\}$, m_j est une flèche élémentaire.
Pour tout $j \in \{1, \ldots n\}$, on a

- Si m_j n'est pas une flèche de $\mathcal{C}_0$, alors $m_j \not\equiv \mathsf{id}_X$, sinon nécessairement $n \geq 2$ et donc $\mathcal{N}(m)$ n'est pas irréductible.
- $m_j \not\equiv \mathsf{j}_X$, car d'après le lemme 3.7, $\mathrm{Arr}(\mathcal{C}_i)(A, \emptyset) = \emptyset$.

Par conséquent, pour tout $j \in \{1, \ldots n\}$, l'une des conditions suivantes est vérifiée

1. $m_j \in \mathrm{Arr}(\mathcal{C}_0)(X, Y)$;
2. $m_j \equiv \&_k(A_0, A_1, A_2, f_1, f_2)$, avec $k \in \{1, 2\}$;
3. $m_j \equiv \mathsf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, f_1', f_2')$.

Par l'absurde, supposons qu'il existe j tel que $m_j \notin \mathrm{Arr}(\mathcal{C}_0)(X, Y)$. Soit j_0 le plus petit j qui satisfait cette propriété. On a donc $m_{j_0} \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$, avec $X \in \mathrm{Obj}(\mathcal{C}_0)$. Par conséquent

$$m_{j_0} \equiv \&_k(A_0, A_1, A_2, f_1, f_2).$$

Si $m_{j_0+1} \equiv \mathsf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, f_1', f_2')$, alors $\mathcal{N}(m)$ n'est pas irréductible, donc

$$m_{j_0+1} \equiv \&_{k'}(A_0', A_1', A_2', f_1', f_2').$$

Finalement, on obtient

$$m_n \equiv \&_{k''}(A_0'', A_1'', A_2'', f_1'', f_2'').$$

ce qui est en contradiction avec $B \in \mathrm{Obj}(\mathcal{C}_0)$. Par conséquent, pour tout $j \in \{1, \ldots n\}$, m_j est une flèche de $\mathcal{C}_0$, et donc $\mathcal{N}(m) \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$. □

Théorème 3.3 *(Forme normale (2))*
Soit $A \in \mathrm{Obj}(\mathcal{C}_0)$, $B \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2) \in \mathrm{Obj}(\mathcal{C}_i)$, $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *Posons*

$$\mathcal{N}(m) \equiv m_n \circ m_{n-1} \circ \cdots \circ m_1,$$

avec $n \geq 1$ *et* $\forall j \in \{1, \ldots n\}$, m_j *est une flèche élémentaire. Alors,*

1. $m_n \equiv \&_k(A_0, A_1, A_2, f_1, f_2)$, *avec* $k \in \{1, 2\}$;
2. *si* $n \geq 2$, *alors* $m_{n-1} \circ \cdots \circ m_1 \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, A_k)$.

Preuve. Similaire à la preuve du théorème 3.2. La preuve complète est développée en appendice A.1. □

Théorème 3.4 *(Forme normale (3))*
Soit $A \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2) \in \mathrm{Obj}(\mathcal{C}_i)$, $B \in \mathrm{Obj}(\mathcal{C}_0)$ *et* $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *Posons*

$$\mathcal{N}(m) \equiv m_n \circ m_{n-1} \circ \cdots \circ m_1,$$

avec $n \geq 1$ *et* $\forall j \in \{1, \ldots n\}$, m_j *est une flèche élémentaire. Alors*

$$m_1 \equiv \mathsf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2).$$

Preuve. Similaire à la preuve du théorème 3.2. La preuve complète est développée en appendice A.2. □

La forme normale d'une flèche n'est pas unique relativement à l'équivalence dans $\mathcal{C}_i$. Dans le cas général,

$$m = m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y) \not\Rightarrow \mathcal{N}(m) \equiv \mathcal{N}(m').$$

Par exemple, on a

$$\&_1(A, B, C, f, g) \circ f = \&_2(A, B, C, f, g) \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A, \mathsf{push}\,(A, B, C, f, g)).$$

Ces deux flèches sont irréductibles et équivalentes dans $\mathcal{C}_i$, mais ne sont pas identiques. Par contre, on a

$$\forall X, Y \in \mathrm{Obj}(\mathcal{C}_0),\ m = m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y) \Rightarrow \mathcal{N}(m) = \mathcal{N}(m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Y).$$

Pour montrer ce résultat, nous démontrons le théorème suivant.

Théorème 3.5 *("Unicité" de la forme normale)*
Soit $m, m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$, *tels que* $m = m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$.

1. *Si* $X, Y \in \mathrm{Obj}(\mathcal{C}_0)$, *alors*
$$\mathcal{N}(m) = \mathcal{N}(m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Y).$$

2. *Si* $X \in \mathrm{Obj}(\mathcal{C}_0)$, $Y \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2)$, *alors*
$\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_i)(Y, Z)$, *on a*
$$\mathcal{N}(h \circ m) = \mathcal{N}(h \circ m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

3. *Si* $X \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2)$, $Y \in \mathrm{Obj}(\mathcal{C}_0)$, *alors*
$\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_i)(W, X)$, *on a*
$$\mathcal{N}(m \circ p) = \mathcal{N}(m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y).$$

4. *Si* $X \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2)$, $Y \equiv \mathsf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$, *alors*
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_i)(Y, Z)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_i)(W, X)$, *on a*
$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Preuve. On prouve les points 1 à 4 par induction sur i. Pour $i = 0$, le point 1 est évident, et il n'y a rien à prouver pour les points 2 à 4. Pour $i+1$, on doit faire une induction sur la longueur de $\mathcal{N}(h)$ pour prouver les points 2 et 4. On prouve alors les points 1, 2, 3, 4 en parallèle, par induction sur la longueur de la preuve que

$$m = m' \in \mathrm{Arr}(\mathcal{C}_{i+1})(X, Y).$$

Cette preuve est détaillée en Appendice A.3. □

3.5 Extensions conservatrices

La suite de précatégories $\mathcal{C}_i$ satisfait deux propriétés importantes: on n'introduit pas de nouvelles flèches entre des objets de $\mathcal{C}_0$, et on n'introduit pas de nouvelles équivalences entre des flèches de $\mathcal{C}_0$. On dit que $\mathcal{C}_i$ est une *extension conservatrice* de $\mathcal{C}_0$.

Pour tout $i \leq j$, on définit un préfoncteur $J_{i,j} : \mathcal{C}_i \to \mathcal{C}_j$ par:

$$\begin{array}{rccc} J_{i,j} : & \mathcal{C}_i & \to & \mathcal{C}_j \\ & A & \mapsto & A \\ & f & \mapsto & f \end{array}$$

$J_{i,j}$ est bien un préfoncteur car d'après le lemme 3.5,

- $A \in \mathrm{Obj}(\mathcal{C}_i) \Rightarrow A \in \mathrm{Obj}(\mathcal{C}_j)$;
- $f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \Rightarrow f \in \mathrm{Arr}(\mathcal{C}_j)(A, B)$;
- $f = f' \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \Rightarrow f = f' \in \mathrm{Arr}(\mathcal{C}_j)(A, B)$.

Théorème 3.6 *Pour tout entier j, la précatégorie $\mathcal{C}_j$ est une extension conservatrice de $\mathcal{C}_0$. Autrement dit, le préfoncteur $J_{0,j}$ est plein et fidèle.*

Ce théorème est équivalent à:

$\forall j \in \mathbb{N}$, $\forall A, B \in \mathrm{Obj}(\mathcal{C}_0)$,

1. $\forall h \in \mathrm{Arr}(\mathcal{C}_j)(A, B)$, $\exists h' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$; $h = h' \in \mathrm{Arr}(\mathcal{C}_j)(A, B)$;
2. $\forall h', h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$,

$$h' = h'' \in \mathrm{Arr}(\mathcal{C}_j)(A, B) \Rightarrow h' = h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B).$$

Preuve. C'est une conséquence immédiate des théorèmes 3.1, 3.2, et 3.5.

1. Il suffit de prendre $h' \equiv \mathcal{N}(h)$. D'après le théorème 3.2, $\mathcal{N}(h) \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$.

2. $h' = h'' \in \mathrm{Arr}(\mathcal{C}_j)(A, B)$, donc d'après le théorème 3.5,

$$\mathcal{N}(h') = \mathcal{N}(h'') \in \mathrm{Arr}(\mathcal{C}_0)(A, B).$$

   De plus, d'après le théorème 3.1, on a

$$\mathcal{N}(h') = h' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$$
$$\mathcal{N}(h'') = h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$$

   d'où le résultat.

□

Remarque 3.3 En général, pour $1 \leq i < j$, $\mathcal{C}_j$ n'est pas une extension conservatrice de $\mathcal{C}_i$. En effet :

1. Le préfoncteur $J_{i,j}$ n'est pas plein, car la règle (10) peut introduire dans la précatégorie $\mathcal{C}_{i+1}$ des nouvelles flèches entre des objets de $\mathcal{C}_i$. Dans l'exemple 3.1, u est une flèche entre deux objets de $\mathcal{C}_1$ qui fait partie de $\mathcal{C}_2$, mais pas de $\mathcal{C}_1$.

2. Le préfoncteur $J_{i,j}$ n'est pas fidèle, car deux flèches de $\mathcal{C}_i$ peuvent être équivalentes à une troisième flèche de $\mathcal{C}_{i+1}$ (et donc, par transitivité, équivalentes dans $\mathcal{C}_{i+1}$), sans être équivalentes dans $\mathcal{C}_i$. On peut donc introduire dans $\mathcal{C}_{i+1}$ des nouvelles équivalences entre flèches de $\mathcal{C}_i$.

3.6 Précatégorie TERME($\mathcal{C}_0$)

Nous pouvons maintenant définir la précatégorie des termes, qui va nous servir de syntaxe pour les constructions de colimites. Les objets de cette précatégorie sont les éléments de $\mathrm{Obj}(\mathcal{C}_i)$, pour i quelconque, et les flèches entre deux objets A et B sont les éléments de $\mathrm{Arr}(\mathcal{C}_j)(A, B)$, pour tous les j tels que $A, B \in \mathrm{Obj}(\mathcal{C}_j)$. De plus, deux flèches sont équivalentes si et seulement si elles sont équivalentes dans un des ensembles $\mathrm{Arr}(\mathcal{C}_j)(A, B)$. On définit donc la précatégorie TERME($\mathcal{C}_0$) comme la "limite" de la suite de précatégories $\mathcal{C}_i$.

Définition 3.12 (Précatégorie TERME($\mathcal{C}_0$))
La précatégorie des termes TERME($\mathcal{C}_0$) est définie de la façon suivante.

- L'ensemble des objets est

$$\mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0)) = \bigcup_{i=0}^{\infty} \mathrm{Obj}(\mathcal{C}_i).$$

- Soit $A, B \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$. Il existe $k \in \mathbb{N}$ tel que $A, B \in \mathrm{Obj}(\mathcal{C}_k)$. L'ensemble des flèches de A vers B est

$$\mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B) = \bigcup_{i=k}^{\infty} \mathrm{Arr}(\mathcal{C}_i)(A, B).$$

- Soit $A, B \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$. Il existe $k \in \mathbb{N}$ tel que $A, B \in \mathrm{Obj}(\mathcal{C}_k)$. La relation d'équivalence sur $\mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ est

$$\mathcal{R}(\mathrm{TERME}(\mathcal{C}_0))(A, B) = \bigcup_{i=k}^{\infty} \mathcal{R}(\mathcal{C}_i)(A, B).$$

D'après le lemme 3.5, ces ensembles sont bien définis. En particulier, la définition des ensembles $\mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ et $\mathcal{R}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ est indépendante de l'entier k choisi.

Proposition 3.2 $\mathrm{TERME}(\mathcal{C}_0)$ *est une précatégorie.*

Soit $J : \mathcal{C}_0 \to \mathrm{TERME}(\mathcal{C}_0)$ le préfoncteur défini par

$$\begin{array}{rccc} J: & \mathcal{C}_0 & \to & \mathrm{TERME}(\mathcal{C}_0) \\ & A & \mapsto & A \\ & f & \mapsto & f. \end{array}$$

Théorème 3.7 *La précatégorie* $\mathrm{TERME}(\mathcal{C}_0)$ *est une extension conservatrice de* $\mathcal{C}_0$. *Autrement dit, le préfoncteur* $J : \mathcal{C}_0 \to \mathrm{TERME}(\mathcal{C}_0)$ *est plein et fidèle.*

Par conséquent, dans $\mathrm{TERME}(\mathcal{C}_0)$, on n'ajoute pas de flèche entre des objets de $\mathcal{C}_0$, et on n'ajoute pas d'équivalence entre flèches de $\mathcal{C}_0$. Cela montre la *cohérence hiérarchique* de la construction.

Preuve. D'après le théorème 3.6, $\forall k \in \mathbb{N}$, la précatégorie $\mathcal{C}_k$ est une extension conservatrice de $\mathcal{C}_0$.

1. Soit $A, B \in \mathrm{Obj}(\mathcal{C}_0)$, et $h \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$. Il existe $k \in \mathbb{N}$ tel que $h \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$. Comme $\mathcal{C}_k$ est une extension conservatrice de $\mathcal{C}_0$, il existe $h' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$, tel que $h = h' \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$, et donc

$$h = h' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B).$$

2. Soit $A, B \in \mathrm{Obj}(\mathcal{C}_0)$, et $h', h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$, tels que

$$h' = h'' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B).$$

Il existe $k \in \mathbb{N}$ tel que $h' = h'' \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$. Comme $\mathcal{C}_k$ est une extension conservatrice de $\mathcal{C}_0$,

$$h' = h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B).$$

□

Théorème 3.8 *La précatégorie* TERME($\mathcal{C}_0$) *est finiment pré-cocomplète. Plus précisément,*

1. *l'objet* $\emptyset$ *est un choix d'objet pré-initial dans* TERME($\mathcal{C}_0$) ;
2. *si* $A, B, C \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$, $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ *et* $g \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, C)$, *alors le triplet*

$$(\mathbf{push}\,(A, B, C, f, g),\ \&_1(A, B, C, f, g),\ \&_2(A, B, C, f, g))$$

est un choix de pré-somme amalgamée de B *et* C *par rapport à* f *et* g *dans* TERME($\mathcal{C}_0$).

Preuve.

1. L'objet $\emptyset$ est pré-initial. Soit $A \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$. Il existe $k \geq 1$ tel que $A \in \mathrm{Obj}(\mathcal{C}_k)$. Donc on a une flèche $j_A \in \mathrm{Arr}(\mathcal{C}_k)(\emptyset, A) \subseteq \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(\emptyset, A)$. Soit deux flèches $u, v \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(\emptyset, A)$. Il existe $k' \geq 1$ tel que $u, v \in \mathrm{Arr}(\mathcal{C}_{k'})(\emptyset, A)$, donc d'après la règle (19), $u = v \in \mathrm{Arr}(\mathcal{C}_{k'})(\emptyset, A)$ et donc $u = v \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(\emptyset, A)$.
2. La preuve est similaire.

□

Considérons une précatégorie $\mathcal{E}$ ayant un objet pré-initial choisi $\emptyset^{\mathcal{E}}$ et des pré-sommes amalgamées choisies. Si A, B et C sont trois objets de $\mathcal{E}$, et $f \in \mathrm{Arr}(\mathcal{E})(A, B)$, $g \in \mathrm{Arr}(\mathcal{E})(A, C)$ deux flèches de $\mathcal{E}$, on note

$$(push^{\mathcal{E}}(A, B, C, f, g),\ \&_1^{\mathcal{E}}(A, B, C, f, g),\ \&_2^{\mathcal{E}}(A, B, C, f, g))$$

la pré-somme amalgamée choisie de B et C par rapport à f et g dans $\mathcal{E}$.
On suppose de plus que, pour tout objet A de $\mathcal{E}$, on a un choix de flèche

$$j_A^{\mathcal{E}} \in \mathrm{Arr}(\mathcal{E})(\emptyset^{\mathcal{E}}, A).$$

De même, si $A, B, C, D \in \mathrm{Obj}(\mathcal{E})$, $f \in \mathrm{Arr}(\mathcal{E})(A, B)$, $g \in \mathrm{Arr}(\mathcal{E})(A, C)$, $f' \in \mathrm{Arr}(\mathcal{E})(B, D)$, $g' \in \mathrm{Arr}(\mathcal{E})(C, D)$, tel que $f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{E})(A, D)$, on a un choix de flèche

$$up^{\mathcal{E}}(A, B, C, D, f, g, f', g') \in \mathrm{Arr}(\mathcal{E})(push^{\mathcal{E}}(A, B, C, f, g), D)$$

telle que

$$up^{\mathcal{E}}(A, B, C, D, f, g, f', g') \circ \&_1^{\mathcal{E}}(A, B, C, f, g) = f' \in \mathrm{Arr}(\mathcal{E})(B, D)$$
$$up^{\mathcal{E}}(A, B, C, D, f, g, f', g') \circ \&_2^{\mathcal{E}}(A, B, C, f, g) = g' \in \mathrm{Arr}(\mathcal{E})(C, D).$$

Théorème 3.9 *(La précatégorie* TERME($\mathcal{C}_0$) *est librement engendrée par objet pré-initial choisi et pré-sommes amalgamées choisies)*

Soit un préfoncteur $F : \mathcal{C}_0 \to \mathcal{E}$. *Alors il existe un unique préfoncteur*

$$G : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{E}$$

tel que

- $G \circ J \equiv F$;
- $G(g \circ f) \equiv G(g) \circ G(f)$;
- $G(\mathsf{id}_A) \equiv id_{G(A)}$;
- $G(\emptyset) \equiv \emptyset^{\mathcal{E}}$;
- $G(\mathsf{j}_A) \equiv j^{\mathcal{E}}_{G(A)}$;
- $G(\mathsf{push}\,(A, B, C, f, g)) \equiv push^{\mathcal{E}}(G(A), G(B), G(C), G(f), G(g))$;
- $G(\&_1(A, B, C, f, g)) \equiv \&^{\mathcal{E}}_1(G(A), G(B), G(C), G(f), G(g))$;
- $G(\&_2(A, B, C, f, g)) \equiv \&^{\mathcal{E}}_2(G(A), G(B), G(C), G(f), G(g))$;
- $G(\mathsf{up}\,(A, B, C, D, f, g, f', g')$
  $\equiv up^{\mathcal{E}}(G(A), G(B), G(C), G(D), G(f), G(g), G(f'), G(g'))$.

La précatégorie $\mathrm{TERME}(\mathcal{C}_0)$ est donc *librement engendrée par objet pré-initial choisi et pré-sommes amalgamées choisies.* Intuitivement cela signifie que la précatégorie $\mathrm{TERME}(\mathcal{C}_0)$ satisfait les propriétés suivantes.

1. Toutes les constructions (de pré-colimites sélectionnées, c'est-à-dire utilisant l'objet pré-initial et les pré-sommes amalgamées) sont distinctes.
2. $\mathrm{TERME}(\mathcal{C}_0)$ ne contient pas d'objets ou flèches autres que ceux de $\mathcal{C}_0$ et ceux obtenus par les constructions de pré-colimites sélectionnées.

Ces deux propriétés correspondent respectivement à l'existence et l'unicité du préfoncteur G.

Preuve. On construit un préfoncteur $G : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{E}$ par induction sur la structure des objets et flèches de $\mathrm{TERME}(\mathcal{C}_0)$. On montre ensuite que ce préfoncteur est unique. La preuve est détaillée en Appendice A.4. □

Ce théorème permet d'*interpréter*, c'est-à-dire de donner une sémantique aux termes. Une précatégorie d'interprétation est une précatégorie avec un objet pré-initial choisi et des pré-sommes amalgamées choisies. À partir d'une interprétation de $\mathcal{C}_0$, c'est-à-dire un préfoncteur

$$\mathcal{S}_0 : \mathcal{C}_0 \to \mathcal{E}$$

on obtient une interprétation des termes, c'est-à-dire un préfoncteur

$$\mathcal{S} : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{E}.$$

Cette précatégorie $\mathcal{E}$ peut par exemple être la catégorie **Spec** (considérée comme une précatégorie, et accompagnée d'un choix d'objet pré-initial et de choix de présommes amalgamées), ou une catégorie de classes d'algèbres permettant d'interpréter les spécifications de base. Dans le chapitre 4, la précatégorie d'interprétation considérée est DIAGR($\mathcal{C}_0$). L'interprétation des objets et flèches de $\mathcal{C}_0$ est donnée par le préfoncteur $I_{\mathcal{C}_0} : \mathcal{C}_0 \to \text{DIAGR}(\mathcal{C}_0)$. D'après le théorème 3.9, cette interprétation peut être prolongée sur les termes en un préfoncteur

$$\mathcal{D} : \text{TERME}(\mathcal{C}_0) \to \text{DIAGR}(\mathcal{C}_0).$$

C'est ce préfoncteur qui va nous permettre, dans le chapitre 4, d'associer à tout terme un diagramme.

3.7 Catégorie Terme($\mathcal{C}_0$)

La catégorie Terme($\mathcal{C}_0$) est la catégorie associée à la précatégorie TERME($\mathcal{C}_0$).

Définition 3.13 (Catégorie des termes) La catégorie Terme($\mathcal{C}_0$) est définie de la façon suivante.

- Les objets de Terme($\mathcal{C}_0$) sont les objets de TERME($\mathcal{C}_0$).
- Pour toute flèche $f \in \text{Arr}(\text{TERME}(\mathcal{C}_0))(A, B)$, $[f] : A \to B$, classe d'équivalence de f modulo $\mathcal{R}(\text{TERME}(\mathcal{C}_0))(A, B)$, est une flèche de Terme($\mathcal{C}_0$).

On a un *foncteur* $J : \mathcal{C}_0 \to \text{Terme}(\mathcal{C}_0)$ associé au préfoncteur $J : \mathcal{C}_0 \to \text{TERME}(\mathcal{C}_0)$.

Théorème 3.10 *La catégorie* Terme($\mathcal{C}_0$) *est une extension conservatrice de* $\mathcal{C}_0$. *Autrement dit, le foncteur* $J : \mathcal{C}_0 \to \text{Terme}(\mathcal{C}_0)$ *est plein et fidèle.*

Preuve. D'après le théorème 3.7, le préfoncteur $J : \mathcal{C}_0 \to \text{TERME}(\mathcal{C}_0)$ est plein et fidèle, donc le foncteur associé $J : \mathcal{C}_0 \to \text{Terme}(\mathcal{C}_0)$ est également plein et fidèle. □

Théorème 3.11 Terme($\mathcal{C}_0$) *est une catégorie finiment cocomplète. Plus précisément,*

1. *l'objet* $\emptyset$ *est initial dans* Terme($\mathcal{C}_0$) ;
2. *si* A, B *et* C *sont des objets de* Terme($\mathcal{C}_0$), $[f] : A \to B$ *et* $[g] : A \to C$ *sont des flèches de* Terme($\mathcal{C}_0$), *alors le triplet*

$$(\textbf{push}\,(A, B, C, f, g),\ [\&_1(A, B, C, f, g)],\ [\&_2(A, B, C, f, g)])$$

est une somme amalgamée de B *et* C *par rapport à* $[f]$ *et* $[g]$ *dans* Terme($\mathcal{C}_0$).

Preuve. Application des lemmes 3.3 et 3.4. □

La catégorie Terme($\mathcal{C}_0$), bien que finiment cocomplète, n'a pas de sommes amalgamées *choisies*. En effet, si f' est un autre représentant de $[f]$, et g' un autre représentant de $[g]$, c'est-à-dire si $[f] = [f']$ et $[g] = [g']$ dans Terme($\mathcal{C}_0$), alors les triplets

$$(\textbf{push}\ (A, B, C, f, g),\ [\&_1(A, B, C, f, g)],\ [\&_2(A, B, C, f, g)])$$
$$(\textbf{push}\ (A, B, C, f', g'),\ [\&_1(A, B, C, f', g')],\ [\&_2(A, B, C, f', g')])$$

sont deux *choix différents* de somme amalgamée de B et C par rapport aux flèches $[f]$ et $[g]$.

Par conséquent, la catégorie Terme($\mathcal{C}_0$) n'est pas *librement engendrée par objet initial choisi et sommes amalgamées choisies*. Il y a en effet "trop de choix de sommes amalgamées" dans Terme($\mathcal{C}_0$). Intuitivement, les constructeurs **push**, $\&_1$ et $\&_2$ dépendent des représentants choisis pour $[f]$ et $[g]$, et donc les *choix* de pré-colimites de TERME($\mathcal{C}_0$) ne sont pas conservés en passant au quotient. Par contre, on peut retrouver une catégorie *librement engendrée par objet initial choisi et sommes amalgamées choisies* en "fusionnant les choix multiples de sommes amalgamées". Cette construction est décrite dans le paragraphe suivant.

3.8 Catégorie libre $\mathcal{L}(\mathcal{C}_0)$

Dans ce paragraphe, nous construisons, à partir de la précatégorie TERME($\mathcal{C}_0$), une catégorie $\mathcal{L}(\mathcal{C}_0)$ librement engendrée par objet initial choisi et sommes amalgamées choisies. L'idée est de définir une relation d'équivalence à la fois sur les objets et les flèches de TERME($\mathcal{C}_0$), afin de fusionner les choix multiples de sommes amalgamées. La relation d'équivalence sur les flèches est plus forte que celle de TERME($\mathcal{C}_0$), c'est-à-dire vérifie

$$f = g \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B) \ \Rightarrow\ f \sim g.$$

Cette construction est inspirée de la "réécriture d'objets" proposée par F. Cury [Cur91].

Nous commençons par définir la relation d'équivalence $\sim$ sur Obj(TERME($\mathcal{C}_0$)). Deux objets équivalents sont isomorphes, mais deux objets isomorphes ne sont pas nécessairement équivalents. Nous définissons d'abord une sous-précatégorie Iso($\mathcal{C}_0$) de TERME($\mathcal{C}_0$). La précatégorie Iso($\mathcal{C}_0$) a pour objets les objets de TERME($\mathcal{C}_0$), et pour flèches *certains isomorphismes* de TERME($\mathcal{C}_0$). Ces isomorphismes vont correspondre aux équivalences entre objets de TERME($\mathcal{C}_0$).

Définition 3.14 (Précatégorie Iso($\mathcal{C}_0$))

- L'ensemble des objets de Iso($\mathcal{C}_0$) est l'ensemble des objets de TERME($\mathcal{C}_0$).

- L'ensemble des flèches de $\mathrm{Iso}(\mathcal{C}_0)$ est défini inductivement de la façon suivante.
  - Pour tout objet A de $\mathrm{TERME}(\mathcal{C}_0)$,
    $$\mathsf{id}_A \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, A).$$
  - Soit $f \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B)$ et $g \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(B, C)$. Alors,
    $$g \circ f \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, C).$$
  - Soit six objets $A_1, B_1, C_1, A_2, B_2, C_2$ de $\mathrm{TERME}(\mathcal{C}_0)$, et trois flèches de $\mathrm{Iso}(\mathcal{C}_0)$
    $$\phi_A \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A_1, A_2),$$
    $$\phi_B \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(B_1, B_2),$$
    $$\phi_C \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(C_1, C_2).$$
    Soit quatre flèches de $\mathrm{TERME}(\mathcal{C}_0)$
    $$f_1 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_1, B_1),$$
    $$g_1 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_1, C_1),$$
    $$f_2 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_2, B_2),$$
    $$g_2 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_2, C_2),$$
    telles que
    $$\phi_B \circ f_1 = f_2 \circ \phi_A \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_1, B_2),$$
    $$\phi_C \circ g_1 = g_2 \circ \phi_A \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A_1, C_2).$$
    Il existe alors une flèche
    $$\mathbf{up}\,(A_1, B_1, C_1, \mathbf{push}\,(A_2, B_2, C_2, f_2, g_2), f_1, g_1, \\ \&_1(A_2, B_2, C_2, f_2, g_2) \circ \phi_B, \&_2(A_2, B_2, C_2, f_2, g_2) \circ \phi_C)$$
    de $\mathbf{push}\,(A_1, B_1, C_1, f_1, g_1)$ vers $\mathbf{push}\,(A_2, B_2, C_2, f_2, g_2)$. Cette flèche, que l'on note
    $$iso(f_1, g_1, f_2, g_2, \phi_A, \phi_B, \phi_C)$$
    est, par définition, une flèche de la précatégorie $\mathrm{Iso}(\mathcal{C}_0)$.
- Soit deux flèches $f, g \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B)$. Par définition, on pose
  $$f = g \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B) \Leftrightarrow f = g \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B).$$

$\mathrm{Iso}(\mathcal{C}_0)$ est évidemment une précatégorie. De plus, toutes les flèches de $\mathrm{Iso}(\mathcal{C}_0)$ sont des isomorphismes dans $\mathrm{TERME}(\mathcal{C}_0)$, et également dans $\mathrm{Iso}(\mathcal{C}_0)$. Pour montrer ce résultat, on définit une fonction *inv* qui associe à toute flèche de $\mathrm{Iso}(\mathcal{C}_0)$ une inverse dans $\mathrm{Iso}(\mathcal{C}_0)$.

Définition 3.15 (Fonction $inv : \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B) \to \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(B, A)$)
La fonction *inv*, qui associe à toute flèche de $\mathrm{Iso}(\mathcal{C}_0)$ de A vers B une flèche de $\mathrm{Iso}(\mathcal{C}_0)$ de B vers A, est définie inductivement sur la structure des flèches de $\mathrm{Iso}(\mathcal{C}_0)$ de la façon suivante.

- $inv(\mathsf{id}_A) = \mathsf{id}_A$;

- $inv(g \circ f) = inv(f) \circ inv(g)$;
- $inv(iso(f_1, g_1, f_2, g_2, \phi_A, \phi_B, \phi_C))$
  $= iso(f_2, g_2, f_1, g_1, inv(\phi_A), inv(\phi_B), inv(\phi_C))$.

On vérifie sans difficulté que, pour toute flèche $f \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B)$, on a bien

- $inv(f) \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(B, A)$;
- $inv(f) \circ f = \mathsf{id}_A \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, A)$;
- $f \circ inv(f) = \mathsf{id}_B \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(B, B)$.

Nous pouvons maintenant définir la relation d'équivalence sur les objets de $\mathrm{TERME}(\mathcal{C}_0)$.

Définition 3.16 (Relation d'équivalence $\sim$ sur les objets de $\mathrm{TERME}(\mathcal{C}_0)$)
Deux objets A et B de $\mathrm{TERME}(\mathcal{C}_0)$ sont équivalents si et seulement si il existe un isomorphisme de A vers B dans $\mathrm{Iso}(\mathcal{C}_0)$.

Notation. Si A et B sont équivalents, on note $A \sim B$, ou, pour préciser l'isomorphisme ϕ de A vers B, $\phi \vdash A \sim B$. Intuitivement, on peut considérer la flèche $\phi : A \to B$ comme la preuve que $A \sim B$.

Lemme 3.11 *La relation $\sim$ est une relation d'équivalence sur* $\mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$.

Preuve. La relation $\sim$ est

- réflexive car $\mathsf{id}_A \vdash A \sim A$;
- symétrique car $\phi \vdash A \sim B \Rightarrow inv(\phi) \vdash B \sim A$;
- transitive car $\phi_1 \vdash A \sim B$, $\phi_2 \vdash B \sim C \Rightarrow \phi_2 \circ \phi_1 \vdash A \sim C$.

□

Soit un objet A de $\mathrm{TERME}(\mathcal{C}_0)$. On note $[[A]]$ la classe d'équivalence de A modulo $\sim$. Autrement dit,

$$[[A]] = \{B \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0)) \,;\, A \sim B\}.$$

À chaque preuve de $A \sim B$ est associé un isomorphisme $\phi \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$. La proposition suivante indique que l'isomorphisme ϕ ne dépend pas de cette preuve.

Proposition 3.3 *Si $\phi_1 \vdash A \sim B$ et $\phi_2 \vdash A \sim B$, alors*

$$\phi_1 = \phi_2 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B).$$

Esquisse de la preuve. On définit un système de réécriture sur les flèches de $\mathrm{Iso}(\mathcal{C}_0)$ de la façon suivante. (Les prémisses qui assurent que tous les termes sont bien formés sont, pour faciliter la lecture, omis.)

Axiomes

- $\phi_3 \circ (\phi_2 \circ \phi_1) \longrightarrow (\phi_3 \circ \phi_2) \circ \phi_1$
- $\phi \circ \mathsf{id}_A \longrightarrow \phi$
- $\mathsf{id}_A \circ \phi \longrightarrow \phi$
- $iso(f_2, g_2, f_3, g_3, \psi_A, \psi_B, \psi_C) \circ iso(f_1, g_1, f_2, g_2, \phi_A, \phi_B, \phi_C)$
  $\longrightarrow iso(f_1, g_1, f_3, g_3, \psi_A \circ \phi_A, \psi_B \circ \phi_B, \psi_C \circ \phi_C)$

Règles de contexte

- $\phi \longrightarrow \phi',\ \psi \longrightarrow \psi' \ \Rightarrow\ \psi \circ \phi \longrightarrow \psi' \circ \phi'$
- $\phi_A \longrightarrow \phi'_A,\ \phi_B \longrightarrow \phi'_B,\ \phi_C \longrightarrow \phi'_C$
  $\Rightarrow\ iso(f_1, g_1, f_2, g_2, \phi_A, \phi_B, \phi_C) \longrightarrow iso(f_1, g_1, f_2, g_2, \phi'_A, \phi'_B, \phi'_C)$

On montre alors successivement les points suivants.

1. Si $\phi \vdash A \sim B$ et $\phi \longrightarrow^* \phi'$, alors $\phi' \in \mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B)$.
2. Le système de réécriture est cohérent. Si $\phi \vdash A \sim B$ et $\phi \longrightarrow^* \phi'$, alors $\phi = \phi' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$.
3. Le système de réécriture est confluent.
4. Le système de réécriture termine.
5. D'après 3 et 4, toute flèche ϕ de $\mathrm{Arr}(\mathrm{Iso}(\mathcal{C}_0))(A, B)$ a donc une forme normale notée $\mathcal{V}(\phi)$. D'après 2, on a $\phi = \mathcal{V}(\phi) \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$.
6. On montre, par induction sur la structure de A et B, que
$$\phi_1 \vdash A \sim B,\ \phi_2 \vdash A \sim B \ \Rightarrow\ \mathcal{V}(\phi_1) \equiv \mathcal{V}(\phi_2).$$
7. On déduit immédiatement de 5 et 6 que
$$\phi_1 \vdash A \sim B,\ \phi_2 \vdash A \sim B \ \Rightarrow\ \phi_1 = \phi_2 \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B).$$

□

Remarque 3.4 Si $\phi \vdash A \sim B$, l'isomorphisme ϕ correspond à une preuve que $A \sim B$ et $\mathcal{V}(\phi)$ correspond à la normalisation de cette preuve.

Notation. Si $\phi \vdash A \sim B$, la flèche $\phi : A \to B$ de $\mathrm{Iso}(\mathcal{C}_0)$ est notée $\langle A, B\rangle : A \to B$. Cette notation est justifiée par la proposition 3.3.

À partir de cette relation d'équivalence sur les objets de $\mathrm{TERME}(\mathcal{C}_0)$, nous définissons une relation d'équivalence sur les flèches de $\mathrm{TERME}(\mathcal{C}_0)$.

Définition 3.17 (Relation d'équivalence $\sim$ sur les flèches de $\mathrm{TERME}(\mathcal{C}_0)$)
Soit deux flèches $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ et $f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A', B')$. Les flèches f et f' sont équivalentes si et seulement si

$$A \sim A',\ B \sim B' \ \text{ et } \ f' \circ \langle A, A'\rangle = \langle B, B'\rangle \circ f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B').$$

On note $f \sim f'$.

Lemme 3.12
La relation $\sim$ sur les flèches de $\mathrm{TERME}(\mathcal{C}_0)$ *est une relation d'équivalence.*

Preuve. La relation $\sim$ est

- réflexive car $\langle A, A\rangle = \mathsf{id}_A \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, A)$;
- symétrique car $inv\langle A, B\rangle = \langle B, A\rangle \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(B, A)$;
- transitive car $\langle B, C\rangle \circ \langle A, B\rangle = \langle A, C\rangle \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, C)$.

□

Soit $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$. On note $[[f]]$ la classe d'équivalence de f modulo $\sim$. Autrement dit,

$$[[f]] = \{f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A', B') \,;\, f \sim f'\}.$$

Lemme 3.13 $\forall f, f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$,

$$f = f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B) \ \Rightarrow \ f \sim f'.$$

Preuve. On a $f' \circ \langle A, A\rangle = \langle B, B\rangle \circ f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ car

$$\langle A, A\rangle = \mathsf{id}_A \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, A)$$
$$\langle B, B\rangle = \mathsf{id}_B \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(B, B).$$

□

Nous définissons maintenant la catégorie $\mathcal{L}(\mathcal{C}_0)$. Intuitivement, $\mathcal{L}(\mathcal{C}_0)$ a pour objets les classes d'équivalence modulo $\sim$ d'objets de $\mathrm{TERME}(\mathcal{C}_0)$ et pour flèches les classes d'équivalence modulo $\sim$ de flèches de $\mathrm{TERME}(\mathcal{C}_0)$.

Définition 3.18 (Catégorie $\mathcal{L}(\mathcal{C}_0)$)

- *Objets.* Soit un objet A de TERME$(\mathcal{C}_0)$. Alors, $[[A]]$ est un objet de $\mathcal{L}(\mathcal{C}_0)$.
- *Flèches.* Soit $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$. Alors, $[[f]] : [[A]] \to [[B]]$ est une flèche de $\mathcal{L}(\mathcal{C}_0)$.

  Cette définition est correcte car si

  $$f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B),\ f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A', B') \text{ et } f \sim f',$$

  alors, on a $A \sim A'$ et $B \sim B'$.
- *Composition.* Soit $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$ et $g \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(C, D)$, avec $B \sim C$. On pose $[[g]] \circ [[f]] = [[g \circ \langle B, C\rangle \circ f]]$.

  Cette définition est indépendante des représentants f et g choisis. En effet, soit $f' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A', B')$ et $g' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(C', D')$ tels que $f \sim f'$ et $g \sim g'$. On a les égalités suivantes dans $\mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, D')$:

  $$\begin{array}{lll} \langle D, D'\rangle \circ g \circ \langle B, C\rangle \circ f & & \\ \quad = g' \circ \langle C, C'\rangle \circ \langle B, C\rangle \circ f & (\text{car } g \sim g') \\ \quad = g' \circ \langle B, C'\rangle \circ f & (\text{car } \langle C, C'\rangle \circ \langle B, C\rangle = \langle B, C'\rangle) \\ \quad = g' \circ \langle B', C'\rangle \circ \langle B, B'\rangle \circ f & (\text{car } \langle B', C'\rangle \circ \langle B, B'\rangle = \langle B, C'\rangle) \\ \quad = g' \circ \langle B', C'\rangle \circ f' \circ \langle A, A'\rangle & (\text{car } f \sim f') \end{array}$$

  Par conséquent, $g \circ \langle B, C\rangle \circ f \sim g' \circ \langle B', C'\rangle \circ f'$, et donc

  $$[[g \circ \langle B, C\rangle \circ f]] = [[g' \circ \langle B', C'\rangle \circ f']].$$

  La composition est évidemment associative.
- *Identités.* Soit $[[A]]$ un objet de $\mathcal{L}(\mathcal{C}_0)$. Alors $[[\mathsf{id}_A]] : [[A]] \to [[A]]$ est une flèche identité.

Remarque 3.5 Si $A \sim B$, alors $\mathsf{id}_A \sim \langle A, B\rangle$. Par conséquent, les flèches $\langle A, B\rangle$ de $\mathrm{Iso}(\mathcal{C}_0)$ correspondent à des identités $[[\langle A, B\rangle]] = [[\mathsf{id}_A]] = [[\mathsf{id}_B]]$ dans $\mathcal{L}(\mathcal{C}_0)$.

On a un préfoncteur $[[-]] : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{L}(\mathcal{C}_0)$ défini de la façon suivante :

$$\begin{array}{lccl} [[-]] & : & \mathrm{TERME}(\mathcal{C}_0) & \to & \mathcal{L}(\mathcal{C}_0) \\ & & A & \mapsto & [[A]] \\ & & f & \mapsto & [[f]]. \end{array}$$

C'est bien un préfoncteur d'après le lemme 3.13. Ce préfoncteur est associé à un foncteur $[[-]] : \mathrm{Terme}(\mathcal{C}_0) \to \mathcal{L}(\mathcal{C}_0)$, défini par :

$$\begin{array}{lccl} [[-]] & : & \mathrm{Terme}(\mathcal{C}_0) & \to & \mathcal{L}(\mathcal{C}_0) \\ & & A & \mapsto & [[A]] \\ & & [f] & \mapsto & [[f]]. \end{array}$$

Proposition 3.4 *La catégorie $\mathcal{L}(\mathcal{C}_0)$ est une extension conservatrice de $\mathcal{C}_0$. Autrement dit, le foncteur $[[-]] \circ J : \mathcal{C}_0 \to \mathcal{L}(\mathcal{C}_0)$ est plein et fidèle.*

Preuve. On utilise le fait que le foncteur $J : \mathcal{C}_0 \to \text{Terme}(\mathcal{C}_0)$ est plein et fidèle (théorème 3.10). On remarque également que la classe d'équivalence d'un objet de $\mathcal{C}_0$ ne contient que cet objet. Autrement dit, pour tout objet A de $\mathcal{C}_0$, $[[J(A)]] = \{J(A)\}$. En effet, dans la définition 3.14, on n'introduit pas d'équivalence entre objets de $\mathcal{C}_0$ autres que les identités.

Le foncteur $[[-]] \circ J$ *est plein.*
Soit deux objets A et B de $\mathcal{C}_0$, et une flèche $h : [[J(A)]] \to [[J(B)]]$ de $\mathcal{L}(\mathcal{C}_0)$. Par définition de $[[-]]$, il existe une flèche g de $\text{TERME}(\mathcal{C}_0)$ telle que $[[g]] = h$, qui a pour domaine $J(A)$ et pour codomaine $J(B)$. Comme le foncteur J est plein, il existe donc une flèche $f : A \to B$ de $\mathcal{C}_0$ telle que $J(f) = g$, ce qui implique que $[[J(f)]] = [[g]] = h$.

Le foncteur $[[-]] \circ J$ *est fidèle.*
Soit deux flèches $f, g : A \to B$ de $\mathcal{C}_0$ telles que $[[J(f)]] = [[J(g)]]$.

$$
\begin{aligned}
& [[J(f)]] = [[J(g)]] \\
& \quad \Rightarrow \; J(f) \sim J(g) \\
& \quad \Rightarrow \; \langle J(B), J(B)\rangle \circ J(f) = J(g) \circ \langle J(A), J(A)\rangle \in \text{Arr}(\text{TERME}(\mathcal{C}_0))(A, B) \\
& \quad \Rightarrow \; J(f) = J(g) \in \text{Arr}(\text{TERME}(\mathcal{C}_0))(A, B) \\
& \quad \Rightarrow \; f = g \qquad\qquad \text{(car } J \text{ est fidèle)}
\end{aligned}
$$

□

Théorème 3.12 *La catégorie* $\mathcal{L}(\mathcal{C}_0)$ *est finiment cocomplète. Plus précisément,*

1. *l'objet* $[[\emptyset]]$ *est un objet initial choisi dans* $\mathcal{L}(\mathcal{C}_0)$ *;*
2. *si* $[[A]]$, $[[B]]$ *et* $[[C]]$ *sont des objets de* $\mathcal{L}(\mathcal{C}_0)$, $[[f]] : [[A]] \to [[B]]$ *et* $[[g]] : [[A]] \to [[C]]$ *sont des flèches de* $\mathcal{L}(\mathcal{C}_0)$, *alors le triplet*

$$([[\textsf{push}\,(A, B, C, f, g)]],\ [[\&_1(A, B, C, f, g)]],\ [[\&_2(A, B, C, f, g)]])$$

est un choix de somme amalgamée de $[[B]]$ *et* $[[C]]$ *par rapport à* $[[f]]$ *et* $[[g]]$ *dans* $\mathcal{L}(\mathcal{C}_0)$.

On vérifie facilement que le choix de somme amalgamée ne dépend pas des représentants A, B, C, f et g respectivement choisis pour $[[A]]$, $[[B]]$, $[[C]]$, $[[f]]$ et $[[g]]$. Le triplet

$$([[\textsf{push}\,(A, B, C, f, g)]],\ [[\&_1(A, B, C, f, g)]],\ [[\&_2(A, B, C, f, g)]])$$

est donc bien une somme amalgamée *choisie.*

On montre enfin que la catégorie $\mathcal{L}(\mathcal{C}_0)$ est libre.

Théorème 3.13 *($\mathcal{L}(\mathcal{C}_0)$ est une extension libre) La catégorie $\mathcal{L}(\mathcal{C}_0)$ est librement engendrée par objet initial choisi et sommes amalgamées choisies. Autrement dit:*

Soit $\mathcal{E}$ une catégorie avec un objet initial choisi et des sommes amalgamées choisies. Soit $F : \mathcal{C}_0 \to \mathcal{E}$ un foncteur. Alors il existe un unique foncteur

$$H : \mathcal{L}(\mathcal{C}_0) \to \mathcal{E}$$

qui conserve fortement l'objet initial choisi, les sommes amalgamées choisies, et tel que

$$H \circ [[-]] \circ J = F.$$

Esquisse de la preuve. On considère le foncteur G : Terme($\mathcal{C}_0$) $\to \mathcal{E}$ associé au préfoncteur G : TERME($\mathcal{C}_0$) $\to \mathcal{E}$ du théorème 3.9.

Existence. On montre que

$$\phi \vdash A \sim A' \Rightarrow G(A) = G(A') \text{ et } G(\phi) = id_{G(A)},$$

par induction sur la longueur de la preuve que $\phi \vdash A \sim A'$. On en déduit que

$$f \sim f' \Rightarrow G(f) = G(f').$$

Par conséquent, il existe un unique foncteur $H : \mathcal{L}(\mathcal{C}_0) \to \mathcal{E}$ tel que $H \circ [[-]] = G$.
On a donc $H \circ [[-]] \circ J = G \circ J = F$.
On vérifie que le foncteur H conserve fortement l'objet initial choisi et les sommes amalgamées choisies.

Unicité. Soit $H' : \mathcal{L}(\mathcal{C}_0) \to \mathcal{E}$ un foncteur qui conserve fortement l'objet initial choisi. les sommes amalgamées choisies, et tel que

$$H' \circ [[-]] \circ J = F.$$

D'après le théorème 3.9, on a donc $H \circ [[-]] = H' \circ [[-]]$. Comme $[[-]]$ est surjectif sur les flèches et les objets, on en déduit que $H = H'$.

□

3.9 Conclusion

Nous avons proposé la construction d'une précatégorie TERME($\mathcal{C}_0$) à objet pré-initial choisi et pré-sommes amalgamées choisies pour décrire des constructions de colimites sur une petite catégorie de base $\mathcal{C}_0$. Cette précatégorie de termes constitue une *syntaxe* pour les constructions modulaires.

Nous avons montré que TERME($\mathcal{C}_0$) est une extension conservatrice de $\mathcal{C}_0$ (théorème 3.10), c'est-à-dire que le préfoncteur d'inclusion J de $\mathcal{C}_0$ dans TERME($\mathcal{C}_0$) est plein et fidèle. Cela signifie d'une part que TERME($\mathcal{C}_0$) ne contient pas de flèches entre des objets de $\mathcal{C}_0$ autres que celles contenues dans $\mathcal{C}_0$, et d'autre part que

TERME($\mathcal{C}_0$) ne contient pas d'égalités entre flèches de $\mathcal{C}_0$ autres que celles qui proviennent de $\mathcal{C}_0$.

Nous avons montré que la précatégorie TERME($\mathcal{C}_0$) est librement engendrée par objet pré-initial choisi et pré-sommes amalgamées choisies (théorème 3.9).

En passant au quotient, on obtient une catégorie finiment cocomplète Terme($\mathcal{C}_0$), qui est également une extension conservatrice de $\mathcal{C}_0$ (théorèmes 3.10 et 3.11).

Enfin, nous avons montré comment construire, à partir de TERME($\mathcal{C}_0$), une catégorie $\mathcal{L}(\mathcal{C}_0)$ librement engendrée par objet initial choisi et sommes amalgamées choisies sur $\mathcal{C}_0$ (théorème 3.13).

Annexe A

Preuves sur la syntaxe

A.1 Preuve du théorème de forme normale (2)

Nous montrons le théorème 3.3, page 30.

Théorème 3.3 *(Forme normale (2))*
Soit $A \in \mathrm{Obj}(\mathcal{C}_0)$, $B \equiv \mathsf{push}\,(A_0, A_1, A_2, f_1, f_2) \in \mathrm{Obj}(\mathcal{C}_i)$, $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *Posons*

$$\mathcal{N}(m) \equiv m_n \circ m_{n-1} \circ \cdots \circ m_1,$$

avec $n \geq 1$ *et* $\forall j \in \{1, \ldots n\}$, m_j *est une flèche élémentaire. Alors,*

1. $m_n \equiv \&_k(A_0, A_1, A_2, f_1, f_2)$, *avec* $k \in \{1, 2\}$;
2. *si* $n \geq 2$, *alors* $m_{n-1} \circ \cdots \circ m_1 \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, A_k)$.

Preuve. Nous montrons successivement les points 1 et 2.

1. On a

   - $m_n \notin \mathrm{Arr}(\mathcal{C}_0)(B', B)$, car $B \notin \mathrm{Obj}(\mathcal{C}_0)$.
   - $m_n \not\equiv \mathsf{id}_X$, sinon nécessairement $n \geq 2$ et donc $\mathcal{N}(m)$ n'est pas irréductible.
   - $m_n \not\equiv \mathsf{j}_X$, car $\mathrm{Arr}(\mathcal{C}_j)(A, \emptyset) = \varnothing$.

   On fait une induction sur n. Si $n = 1$, alors comme $A \in \mathrm{Obj}(\mathcal{C}_0)$, on a

   $$m_1 \equiv \&_k(A_0, A_1, A_2, f_1, f_2).$$

   Pas d'induction : par l'absurde, si $m_n \not\equiv \&_k(A_0, A_1, A_2, f_1, f_2)$, alors

   $$m_n \equiv \mathsf{up}\,(A'_0, A'_1, A'_2, B, f'_1, f'_2, g'_1, g'_2) \in \mathrm{Arr}(\mathcal{C}_i)(\mathsf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2), B).$$

   Par hypothèse d'induction,

   $$m_{n-1} \equiv \&_{k'}(A'_0, A'_1, A'_2, f'_1, f'_2),$$

   ce qui est en contradiction avec $\mathcal{N}(m)$ irréductible. On a donc

   $$m_n \equiv \&_k(A_0, A_1, A_2, f_1, f_2).$$

2. Par induction sur n. Si $n = 2$, alors

$$\mathcal{N}(m) \equiv \&_k(A_0, A_1, A_2, f_1, f_2) \circ m_1.$$

D'après le lemme 3.7, $A_k \not\equiv \emptyset$.

Si $A_k \in \mathrm{Obj}(\mathcal{C}_0)$, alors $m_1 \in \mathrm{Arr}(\mathcal{C}_0)(A, A_k)$.

Sinon, $A_k \equiv \mathbf{push}\ (A'_0, A'_1, A'_2, f'_1, f'_2) \in \mathrm{Obj}(\mathcal{C}_{i-1})$, et donc

$$m_1 \equiv \&_{k'}(A'_0, A'_1, A'_2, f'_1, f'_2) \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, A_k).$$

Pas d'induction :

$$\mathcal{N}(m) \equiv \&_k(A_0, A_1, A_2, f_1, f_2) \circ m_{n-1} \circ \cdots \circ m_1 \in \mathrm{Arr}(\mathcal{C}_i)(A, B).$$

D'après le lemme 3.7, $A_k \not\equiv \emptyset$.

Si $A_k \in \mathrm{Obj}(\mathcal{C}_0)$, alors d'après le théorème 3.2,

$$m_{n-1} \circ \cdots \circ m_1 \in \mathrm{Arr}(\mathcal{C}_0)(A, A_k)$$

Si $A_k \equiv \mathbf{push}\ (A'_0, A'_1, A'_2, f'_1, f'_2) \in \mathrm{Obj}(\mathcal{C}_{i-1})$, alors

$$m_{n-1} \equiv \&_{k'}(A'_0, A'_1, A'_2, f'_1, f'_2) \in \mathrm{Arr}(\mathcal{C}_{i-1})(A'_{k'}, A_k)$$

et, par hypothèse d'induction,

$$m_{n-2} \circ \cdots \circ m_1 \in \mathrm{Obj}(\mathcal{C}_{i-2})(A, A'_{k'}),$$

d'où

$$m_{n-1} \circ \cdots \circ m_1 \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, A_k).$$

□

A.2 Preuve du théorème de forme normale (3)

Nous montrons le théorème 3.4, page 31.

Théorème 3.4 *(Forme normale (3))*
Soit $A \equiv \mathbf{push}\ (A_0, A_1, A_2, f_1, f_2) \in \mathrm{Obj}(\mathcal{C}_i)$, $B \in \mathrm{Obj}(\mathcal{C}_0)$ *et* $m \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$. *Posons*

$$\mathcal{N}(m) \equiv m_n \circ m_{n-1} \circ \cdots \circ m_1,$$

avec $n \geq 1$ *et* $\forall j \in \{1, \ldots n\}$, m_j *est une flèche élémentaire. Alors*

$$m_1 \equiv \mathbf{up}\ (A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2).$$

Preuve. D'abord, on a

- $m_1 \notin \mathrm{Arr}(\mathcal{C}_0)(A, A')$ car $A \notin \mathrm{Obj}(\mathcal{C}_0)$.
- $m_1 \not\equiv \mathsf{id}_X$, sinon nécessairement, $n \geq 2$ et donc $\mathcal{N}(m)$ n'est pas irréductible.
- $m_1 \not\equiv \mathsf{j}_X$, car $A \not\equiv \emptyset$.

Par conséquent, on a soit

$$m_1 \equiv \mathsf{up}\,(A_0, A_1, A_2, B, f_1, f_2, g_1, g_2),$$

soit

$$m_1 \equiv \&_{k'}(A'_0, A'_1, A'_2, f'_1, f'_2).$$

On montre par induction sur n que c'est la première condition qui est satisfaite. Si $n = 1$, alors comme $B \in \mathrm{Obj}(\mathcal{C}_0)$, on a

$$m_1 \equiv \mathsf{up}\,(A_0, A_1, A_2, B, f_1, f_2, g_1, g_2).$$

Pas d'induction : par l'absurde, si

$$m_1 \not\equiv \mathsf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2),$$

alors

$$m_1 \equiv \&_{k'}(A'_0, A'_1, A'_2, f'_1, f'_2).$$

Par hypothèse d'induction,

$$m_2 \equiv \mathsf{up}\,(A'_0, A'_1, A'_2, A'_3, f'_1, f'_2, g'_1, g'_2),$$

ce qui est en contradiction avec $\mathcal{N}(m)$ irréductible. Par conséquent,

$$m_1 \equiv \mathsf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2).$$

□

A.3 Preuve du théorème d'"unicité" de la forme normale

Nous montrons le théorème 3.5, page 31.

Théorème 3.5 *("Unicité" de la forme normale)*
Soit $m, m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$, *tels que* $m = m' \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$.

1. *Si* $X, Y \in \mathrm{Obj}(\mathcal{C}_0)$, *alors*

$$\mathcal{N}(m) = \mathcal{N}(m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Y).$$

2. *Si* $X \in \mathrm{Obj}(\mathcal{C}_0)$, $Y \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$, *alors* $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_i)(Y, Z)$, *on a*

$$\mathcal{N}(h \circ m) = \mathcal{N}(h \circ m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

3. *Si* $X \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$, $Y \in \mathrm{Obj}(\mathcal{C}_0)$, *alors* $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_i)(W, X)$, *on a*

$$\mathcal{N}(m \circ p) = \mathcal{N}(m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y).$$

4. *Si* $X \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$, $Y \equiv \mathbf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$, *alors* $\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_i)(Y, Z)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_i)(W, X)$, *on a*

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Preuve. Nous montrons les points 1 à 4 par induction sur i. Pour $i = 0$, le point 1 est évident, et il n'y a rien à prouver pour les points 2 à 4. Pour $i + 1$, on prouve les points 2 et 4 par induction sur la longueur de $\mathcal{N}(h)$. Nous prouvons alors les points 1 à 4 en parallèle, par induction sur la longueur de la preuve que

$$m = m' \in \mathrm{Arr}(\mathcal{C}_{i+1})(X, Y).$$

Il faut considérer les règles (11) à (23).
En réalité, nous avons besoin de l'hypothèse d'induction sur la longueur de $\mathcal{N}(h)$ uniquement pour la règle (20). Dans tous les autres cas, il est inutile de prouver le cas de base, puis le pas d'induction. Les points 2 et 4 sont donc montrés directement, pour toutes les règles sauf la règle (20), et par induction sur la longueur de $\mathcal{N}(h)$ pour la règle (20).

Règle (11) $m = m' \in \mathrm{Arr}(\mathcal{C}_0)(X, Y)$.

1. D'après le théorème 3.1, on a

$$m = \mathcal{N}(m) \in \mathrm{Arr}(\mathcal{C}_0)(X, Y)$$
$$m' = \mathcal{N}(m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Y)$$

et donc $\mathcal{N}(m) = \mathcal{N}(m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Y)$.

Il n'y a rien à démontrer pour les points 2 à 4.

Règle (12) Les 4 points sont évidents.

Règle (13) Les 4 points sont évidents, par symétrie de l'égalité dans $\mathcal{C}_0$.

Règle (14) Les 4 points sont évidents, par transitivité de l'égalité dans $\mathcal{C}_0$.

Règle (15) $m \equiv g \circ f$, $m' \equiv g' \circ f'$, avec $f = f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$ et $g = g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, C)$.

1. $A, C \in \mathrm{Obj}(\mathcal{C}_0)$.
   - $B \in \mathrm{Obj}(\mathcal{C}_0)$. Par hypothèse d'induction, on a
   $$\mathcal{N}(f) = \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \qquad (i)$$

$$\mathcal{N}(g) = \mathcal{N}(g') \in \mathrm{Arr}(\mathcal{C}_0)(B, C) \qquad (ii)$$

$$\begin{array}{lll} \mathcal{N}(g \circ f) & & \\ \quad \equiv & \mathcal{N}(\mathcal{N}(g) \circ \mathcal{N}(f)) \in \mathrm{Arr}(\mathcal{C}_0)(A, C) & \\ \quad = & \mathcal{N}(g) \circ \mathcal{N}(f) \in \mathrm{Arr}(\mathcal{C}_0)(A, C) & \text{(théorème 3.1)} \\ \quad = & \mathcal{N}(g') \circ \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_0)(A, C) & \text{(d'après } (i) \text{ et } (ii)) \\ \quad = & \mathcal{N}(\mathcal{N}(g') \circ \mathcal{N}(f')) \in \mathrm{Arr}(\mathcal{C}_0)(A, C) & \text{(théorème 3.1)} \\ \quad \equiv & \mathcal{N}(g' \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, C) & \end{array}$$

- $B \equiv \textsf{push}\,(A_0, A_1, A_2, f_1, f_2)$. Par hypothèse d'induction, on a :
  $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
  $$\mathcal{N}(h \circ f) = \mathcal{N}(h \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) \qquad (i)$$
  $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,
  $$\mathcal{N}(g \circ p) = \mathcal{N}(g' \circ p) \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, C) \qquad (ii)$$
  Par conséquent,

  $$\begin{array}{lll} \mathcal{N}(g \circ f) & = & \mathcal{N}(g \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, C) \quad ((i), \text{ avec } h \equiv g) \\ & = & \mathcal{N}(g' \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, C) \quad ((ii), \text{ avec } p \equiv f') \end{array}$$

- $B \equiv \emptyset$: impossible d'après le lemme 3.7.

2. $A \in \mathrm{Obj}(\mathcal{C}_0)$, $C \equiv \textsf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$. Il faut montrer que
   $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
   $$\mathcal{N}(h \circ g \circ f) = \mathcal{N}(h \circ g' \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z).$$

   - $B \in \mathrm{Obj}(\mathcal{C}_0)$. Par hypothèse d'induction, on a
     $$\mathcal{N}(f) = \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \qquad (i)$$
     $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
     $$\mathcal{N}(h \circ g) = \mathcal{N}(h \circ g') \in \mathrm{Arr}(\mathcal{C}_0)(B, Z) \qquad (ii)$$
     Par conséquent,

     $$\begin{array}{lll} \mathcal{N}(h \circ g \circ f) & & \\ \quad \equiv & \mathcal{N}(\mathcal{N}(h \circ g) \circ \mathcal{N}(f)) & \\ \quad = & \mathcal{N}(h \circ g) \circ \mathcal{N}(f) \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) & \text{(théorème 3.1)} \\ \quad = & \mathcal{N}(h \circ g') \circ \mathcal{N}(f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) & \text{(d'après } (i) \text{ et } (ii)) \\ \quad = & \mathcal{N}(\mathcal{N}(h \circ g') \circ \mathcal{N}(f')) & \text{(théorème 3.1)} \\ \quad \equiv & \mathcal{N}(h \circ g' \circ f') & \end{array}$$

   - $B \equiv \textsf{push}\,(A_0, A_1, A_2, f_1, f_2)$. Par hypothèse d'induction, on a
     $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,
     $$\mathcal{N}(h' \circ f) = \mathcal{N}(h' \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) \qquad (i)$$
     $\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p' \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, B)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
     $$\mathcal{N}(h \circ g \circ p') = \mathcal{N}(h \circ g' \circ p') \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) \qquad (ii)$$
     Par conséquent,

     $$\begin{array}{lll} \mathcal{N}(h \circ g \circ f) & & \\ \quad = & \mathcal{N}(h \circ g \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) & ((i), \text{ avec } h' \equiv h \circ g) \\ \quad = & \mathcal{N}(h \circ g' \circ f') \in \mathrm{Arr}(\mathcal{C}_0)(A, Z) & ((ii), \text{ avec } p' \equiv f') \end{array}$$

- $B \equiv \emptyset$: impossible d'après le lemme 3.7.

3. $A \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$, $C \in \mathrm{Obj}(\mathcal{C}_0)$. On doit montrer que
$\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,

$$\mathcal{N}(g \circ f \circ p) = \mathcal{N}(g' \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, C).$$

- $B \in \mathrm{Obj}(\mathcal{C}_0)$. Par hypothèse d'induction, on a
$\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,
$$\mathcal{N}(f \circ p) = \mathcal{N}(f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, B) \qquad (i)$$
et
$$\mathcal{N}(g) = \mathcal{N}(g') \in \mathrm{Arr}(\mathcal{C}_0)(B, C) \qquad (ii)$$
Par conséquent,

$$\begin{array}{lll}
\mathcal{N}(g \circ f \circ p) & & \\
\quad \equiv & \mathcal{N}(\mathcal{N}(g) \circ \mathcal{N}(f \circ p)) & \\
\quad = & \mathcal{N}(g) \circ \mathcal{N}(f \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, C) & \text{(théorème 3.1)} \\
\quad = & \mathcal{N}(g') \circ \mathcal{N}(f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, C) & \text{(cf. } (i) \text{ et } (ii)) \\
\quad = & \mathcal{N}(\mathcal{N}(g') \circ \mathcal{N}(f' \circ p)) \in \mathrm{Arr}(\mathcal{C}_0)(W, C) & \text{(théorème 3.1)} \\
\quad \equiv & \mathcal{N}(g' \circ f' \circ p) &
\end{array}$$

- $B \equiv \mathbf{push}\,(A_0', A_1', A_2', f_1', f_2')$. Par hypothèse d'induction, on a
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,
$$\mathcal{N}(h \circ f \circ p) = \mathcal{N}(h \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) \qquad (i)$$
et $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p' \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, B)$,
$$\mathcal{N}(g \circ p') = \mathcal{N}(g' \circ p') \in \mathrm{Arr}(\mathcal{C}_0)(W, C) \qquad (ii)$$
Par conséquent,

$$\begin{array}{lll}
\mathcal{N}(g \circ f \circ p) & & \\
\quad = & \mathcal{N}(g \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, C) & ((i), \text{ avec } h \equiv g) \\
\quad = & \mathcal{N}(g' \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, C) & ((ii), \text{ avec } p' \equiv f' \circ p)
\end{array}$$

- $B \equiv \emptyset$. Il n'existe pas de flèche $p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, sinon, on aurait une flèche $f \circ p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, \emptyset)$, avec $W \in \mathrm{Obj}(\mathcal{C}_0)$. Par conséquent, on a le résultat.

4. $A \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$ et $C \equiv \mathbf{push}\,(A_0'', A_1'', A_2'', f_1'', f_2'')$. On doit montrer que
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,

$$\mathcal{N}(h \circ g \circ f \circ p) = \mathcal{N}(h \circ g' \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

- $B \in \mathrm{Obj}(\mathcal{C}_0)$. Par hypothèse d'induction, on a
$\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,
$$\mathcal{N}(f \circ p) = \mathcal{N}(f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, B) \qquad (i)$$
$\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
$$\mathcal{N}(h \circ g) = \mathcal{N}(h \circ g') \in \mathrm{Arr}(\mathcal{C}_0)(B, Z) \qquad (ii)$$

Par conséquent,

$$\begin{aligned}
&\mathcal{N}(h \circ g \circ f \circ p) \\
&\quad \equiv \mathcal{N}(\mathcal{N}(h \circ g) \circ \mathcal{N}(f \circ p)) \\
&\quad = \mathcal{N}(h \circ g) \circ \mathcal{N}(f \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) && \text{(théorème 3.1)} \\
&\quad = \mathcal{N}(h \circ g') \circ \mathcal{N}(f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) && \text{(d'après } (i) \text{ et } (ii)) \\
&\quad = \mathcal{N}(\mathcal{N}(h \circ g') \circ \mathcal{N}(f' \circ p)) && \text{(théorème 3.1)} \\
&\quad \equiv \mathcal{N}(h \circ g' \circ f' \circ p)
\end{aligned}$$

- $B \equiv \mathsf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$. Par hypothèse d'induction, on a
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, $\forall h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,
$$\mathcal{N}(h' \circ f \circ p) = \mathcal{N}(h' \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) \qquad (i)$$
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p' \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, B)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(C, Z)$,
$$\mathcal{N}(h \circ g \circ p') = \mathcal{N}(h \circ g' \circ p') \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) \qquad (ii)$$
Par conséquent,

$$\begin{aligned}
&\mathcal{N}(h \circ g \circ f \circ p) \\
&\quad = \mathcal{N}(h \circ g \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) && ((i), \text{ avec } h' \equiv h \circ g) \\
&\quad = \mathcal{N}(h \circ g' \circ f' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z) && ((ii), \text{ avec } p' \equiv f' \circ p)
\end{aligned}$$

- $B \equiv \emptyset$. Il n'existe pas de flèche $p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, sinon, on aurait une flèche $f \circ p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, \emptyset)$, avec $W \in \mathrm{Obj}(\mathcal{C}_0)$. Par conséquent, on a le résultat.

Règle (16) $m \equiv (h \circ g) \circ f$ et $m' \equiv h \circ (g \circ f)$.

1. $\mathcal{N}((h \circ g) \circ f) \equiv \mathcal{N}(h \circ (g \circ f))$.

Les points 2 à 4 sont aussi évidents.

Règle (17) $m \equiv f \circ \mathsf{id}_A$ et $m' \equiv f$, avec $f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$.

1. On a
$$\mathcal{N}(m) \equiv \mathcal{N}(m').$$
2. $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,
$$\mathcal{N}(h \circ m) \equiv \mathcal{N}(h \circ m').$$
3. $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,
$$\mathcal{N}(m \circ p) \equiv \mathcal{N}(m' \circ p).$$
4. $\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,
$$\mathcal{N}(h \circ m \circ p) \equiv \mathcal{N}(h \circ m' \circ p).$$

Règle (18) Cas similaire au précédent.

Règle (19) Il n'y a rien à démontrer.

Règle (20) $m \equiv \&_1(A_0, A_1, A_2, f_1, f_2) \circ f_1$ et $m' \equiv \&_2(A_0, A_1, A_2, f_1, f_2) \circ f_2$.

1. Il n'y a rien à démontrer.
2. $X \equiv A_0 \in \mathrm{Obj}(\mathcal{C}_0)$ et $Y \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$. Il faut montrer que $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(Y, Z)$,

$$\mathcal{N}(h \circ m) = \mathcal{N}(h \circ m').$$

Induction sur la longueur de $\mathcal{N}(h)$. Cas de base : $\mathcal{N}(h)$ est de longueur 1. D'après le théorème 3.4,

$$\mathcal{N}(h) \equiv \mathbf{up}\,(A_0, A_1, A_2, Z, f_1, f_2, g_1, g_2)$$

$$\begin{array}{rcl} \mathcal{N}(h \circ m) & \equiv & \mathcal{N}(\mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_1(A_0, \ldots f_2) \circ f_1) \\ & \equiv & \mathcal{N}(g_1 \circ f_1) \\ \mathcal{N}(h \circ m') & \equiv & \mathcal{N}(\mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_2(A_0, \ldots f_2) \circ f_2) \\ & \equiv & \mathcal{N}(g_2 \circ f_2) \end{array}$$

Or, on a $g_1 \circ f_1 = g_2 \circ f_2 \in \mathrm{Arr}(\mathcal{C}_i)(X, Z)$, donc par hypothèse d'induction sur i,

$$\mathcal{N}(g_1 \circ f_1) = \mathcal{N}(g_2 \circ f_2) \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

Par conséquent,

$$\mathcal{N}(h \circ m) = \mathcal{N}(h \circ m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

Pas d'induction : on suppose le résultat démontré pour $\mathcal{N}(h)$ de longueur n, et on le montre pour $\mathcal{N}(h)$ de longueur $n + 1$. D'après le théorème 3.4,

$$\mathcal{N}(h) \equiv h' \circ \mathbf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2).$$

$$\begin{array}{rcl} \mathcal{N}(h \circ m) & \equiv & \mathcal{N}(h' \circ \mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_1(A_0, \ldots f_2) \circ f_1) \\ & \equiv & \mathcal{N}(h' \circ g_1 \circ f_1) \\ \mathcal{N}(h \circ m') & \equiv & \mathcal{N}(h' \circ \mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_2(A_0, \ldots f_2) \circ f_2) \\ & \equiv & \mathcal{N}(h' \circ g_2 \circ f_2) \end{array}$$

Or, on a $g_1 \circ f_1 = g_2 \circ f_2 \in \mathrm{Arr}(\mathcal{C}_i)(X, Y)$, et h' est une forme normale de longueur n, donc par hypothèse d'induction sur n,

$$\mathcal{N}(h' \circ g_1 \circ f_1) = \mathcal{N}(h' \circ g_2 \circ f_2) \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

Par conséquent,

$$\mathcal{N}(h \circ m) = \mathcal{N}(h \circ m') \in \mathrm{Arr}(\mathcal{C}_0)(X, Z).$$

3. Il n'y a rien à démontrer.

4. $X \equiv A_0 \equiv \mathbf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$ et $Y \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$. Il faut montrer que
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, X)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(Y, Z)$,

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p).$$

Remarquons d'abord que $X \in \mathrm{Obj}(\mathcal{C}_i)$, donc d'après le théorème 3.3–2, $\mathcal{N}(p) \in \mathrm{Arr}(\mathcal{C}_i)(W, X)$.
Cas de base sur la longueur de $\mathcal{N}(h)$:

$$\mathcal{N}(h) \equiv \mathbf{up}\,(A_0, A_1, A_2, Z, f_1, f_2, g_1, g_2).$$

$$\begin{array}{rcl} \mathcal{N}(h \circ m \circ p) & \equiv & \mathcal{N}(\mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_1(A_0, \ldots f_2) \circ f_1 \circ p) \\ & \equiv & \mathcal{N}(g_1 \circ f_1 \circ \mathcal{N}(p)) \\ \mathcal{N}(h \circ m' \circ p) & \equiv & \mathcal{N}(\mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_2(A_0, \ldots f_2) \circ f_2 \circ p) \\ & \equiv & \mathcal{N}(g_2 \circ f_2 \circ \mathcal{N}(p)) \end{array}$$

$\mathcal{N}(p) \in \mathrm{Arr}(\mathcal{C}_i)$, $g_1 \circ f_1 = g_2 \circ f_2 \in \mathrm{Arr}(\mathcal{C}_i)(X, Z)$, donc par hypothèse d'induction sur i, on a

$$\mathcal{N}(g_1 \circ f_1 \circ \mathcal{N}(p)) = \mathcal{N}(g_2 \circ f_2 \circ \mathcal{N}(p)) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z)$$

d'où

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Pas d'induction : on suppose le résultat démontré pour $\mathcal{N}(h)$ de longueur n, et on le montre pour $\mathcal{N}(h)$ de longueur $n + 1$.

$$\mathcal{N}(h) \equiv h' \circ \mathbf{up}\,(A_0, A_1, A_2, A_3, f_1, f_2, g_1, g_2)$$

$$\begin{array}{rcl} \mathcal{N}(h \circ m \circ p) & \equiv & \mathcal{N}(h' \circ \mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_1(A_0, \ldots f_2) \circ f_1 \circ p) \\ & \equiv & \mathcal{N}(h' \circ g_1 \circ f_1 \circ p) \\ \mathcal{N}(h \circ m' \circ p) & \equiv & \mathcal{N}(h' \circ \mathbf{up}\,(A_0, \ldots g_1, g_2) \circ \&_2(A_0, \ldots f_2) \circ f_2 \circ p) \\ & \equiv & \mathcal{N}(h' \circ g_2 \circ f_2 \circ p) \end{array}$$

On a $g_1 \circ f_1 = g_2 \circ f_2 \in \mathrm{Arr}(\mathcal{C}_i)(X, Z)$, h' est une forme normale de longueur n, donc par hypothèse d'induction sur n, on a

$$\mathcal{N}(h' \circ g_1 \circ f_1 \circ p) = \mathcal{N}(h' \circ g_2 \circ f_2 \circ p),$$

d'où

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Règle (21) $m \equiv \mathbf{up}\,(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g)$, $m' \equiv f'$.

1. On a

$$\mathcal{N}(m) \equiv \mathcal{N}(m').$$

2. $\forall Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,

$$\mathcal{N}(h \circ m) \equiv \mathcal{N}(h \circ m').$$

3. $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$,

$$\mathcal{N}(m \circ p) \equiv \mathcal{N}(m' \circ p).$$

4. $\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, A)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(B, Z)$,

$$\mathcal{N}(h \circ m \circ p) \equiv \mathcal{N}(h \circ m' \circ p).$$

Règle (22) Cas similaire au cas précédent.

Règle (23) $m = m'$ parce que

$$m \circ \&_1(A_0, A_1, A_2, f_1, f_2) = m' \circ \&_1(A_0, A_1, A_2, f_1, f_2) \in \mathrm{Arr}(\mathcal{C}_{i+1})(A_1, Y),$$
$$m \circ \&_2(A_0, A_1, A_2, f_1, f_2) = m' \circ \&_2(A_0, A_1, A_2, f_1, f_2) \in \mathrm{Arr}(\mathcal{C}_{i+1})(A_2, Y).$$

1. Il n'y a rien à démontrer.
2. Il n'y a rien à démontrer.
3. $X \equiv \mathbf{push}\ (A_0, A_1, A_2, f_1, f_2)$ et $Y \in \mathrm{Obj}(\mathcal{C}_0)$. Il faut montrer que $\forall W \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, X)$,

$$\mathcal{N}(m \circ p) = \mathcal{N}(m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y).$$

Soit $\mathcal{N}(p) \equiv p_n \circ p_{n-1} \circ \cdots \circ p_1$, avec $n \geq 1$. D'après le théorème 3.3,

$$p_n \equiv \&_k(A_0, A_1, A_2, f_1, f_2).$$

Si $n = 1$, on a par hypothèse d'induction

$$\begin{aligned}&\mathcal{N}(m \circ \&_k(A_0, A_1, A_2, f_1, f_2)) \\ &\quad = \mathcal{N}(m' \circ \&_k(A_0, A_1, A_2, f_1, f_2)) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y)\end{aligned}$$

d'où

$$\mathcal{N}(m \circ p) = \mathcal{N}(m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y).$$

Si $n \geq 2$, on a par hypothèse d'induction

$$\begin{aligned}&\mathcal{N}(m \circ \&_k(A_0, A_1, A_2, f_1, f_2) \circ p_{n-1} \circ \cdots \circ p_1) \\ &\quad = \mathcal{N}(m' \circ \&_k(A_0, A_1, A_2, f_1, f_2) \circ p_{n-1} \circ \cdots \circ p_1) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y)\end{aligned}$$

d'où

$$\mathcal{N}(m \circ p) = \mathcal{N}(m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Y).$$

4. $X \equiv \mathbf{push}\,(A_0, A_1, A_2, f_1, f_2)$ et $Y \equiv \mathbf{push}\,(A'_0, A'_1, A'_2, f'_1, f'_2)$. Il faut montrer que
$\forall W, Z \in \mathrm{Obj}(\mathcal{C}_0)$, $\forall p \in \mathrm{Arr}(\mathcal{C}_{i+1})(W, X)$, $\forall h \in \mathrm{Arr}(\mathcal{C}_{i+1})(Y, Z)$,

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Soit $\mathcal{N}(p) \equiv p_n \circ p_{n-1} \circ \cdots \circ p_1$, avec $n \geq 1$. D'après le théorème 3.3,

$$p_n \equiv \&_k(A_0, A_1, A_2, f_1, f_2).$$

Si $n = 1$, on a par hypothèse d'induction

$$\begin{aligned}&\mathcal{N}(h \circ m \circ \&_k(A_0, A_1, A_2, f_1, f_2)) \\ &\quad = \mathcal{N}(h \circ m' \circ \&_k(A_0, A_1, A_2, f_1, f_2)) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z)\end{aligned}$$

d'où

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

Si $n \geq 2$, on a par hypothèse d'induction

$$\begin{aligned}&\mathcal{N}(h \circ m \circ \&_k(A_0, A_1, A_2, f_1, f_2) \circ p_{n-1} \circ \cdots \circ p_1) \\ &\quad = \mathcal{N}(h \circ m' \circ \&_k(A_0, A_1, A_2, f_1, f_2) \circ p_{n-1} \circ \cdots \circ p_1) \\ &\quad \in \mathrm{Arr}(\mathcal{C}_0)(W, Z)\end{aligned}$$

d'où

$$\mathcal{N}(h \circ m \circ p) = \mathcal{N}(h \circ m' \circ p) \in \mathrm{Arr}(\mathcal{C}_0)(W, Z).$$

□

A.4 Preuve du théorème 3.9

Nous montrons le théorème 3.9 page 36.

Théorème 3.9 *(La précatégorie* $\mathrm{TERME}(\mathcal{C}_0)$ *est librement engendrée par objet pré-initial choisi et pré-sommes amalgamées choisies)*
Considérons une précatégorie $\mathcal{E}$ *ayant un objet pré-initial choisi* $\emptyset^{\mathcal{E}}$ *et des pré-sommes amalgamées choisies. Si* A, B, C *sont trois objets de* $\mathcal{E}$, *et* $f \in \mathrm{Arr}(\mathcal{E})(A, B)$, $g \in \mathrm{Arr}(\mathcal{E})(A, C)$ *deux flèches de* $\mathcal{E}$, *on note*

$$(push^{\mathcal{E}}(A, B, C, f, g),\ \&_1^{\mathcal{E}}(A, B, C, f, g),\ \&_2^{\mathcal{E}}(A, B, C, f, g))$$

la pré-somme amalgamée choisie de B *et* C *par rapport à* f *et* g *dans* $\mathcal{E}$.
On suppose de plus que, pour tout objet A *de* $\mathcal{E}$, *on a un choix de flèche*

$$j_A^{\mathcal{E}} \in \mathrm{Arr}(\mathcal{E})(\emptyset^{\mathcal{E}}, A).$$

De même, si $A, B, C, D \in \mathrm{Obj}(\mathcal{E})$, $f \in \mathrm{Arr}(\mathcal{E})(A, B)$, $g \in \mathrm{Arr}(\mathcal{E})(A, C)$, $f' \in \mathrm{Arr}(\mathcal{E})(B, D)$, $g' \in \mathrm{Arr}(\mathcal{E})(C, D)$, *tel que* $f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{E})(A, D)$, *on a un choix de flèche*

$$up^{\mathcal{E}}(A, B, C, D, f, g, f', g') \in \mathrm{Arr}(\mathcal{E})(push^{\mathcal{E}}(A, B, C, f, g), D)$$

telle que

$$up^{\mathcal{E}}(A,B,C,D,f,g,f',g') \circ \&_1^{\mathcal{E}}(A,B,C,f,g) = f' \in \mathrm{Arr}(\mathcal{E})(B,D)$$
$$up^{\mathcal{E}}(A,B,C,D,f,g,f',g') \circ \&_2^{\mathcal{E}}(A,B,C,f,g) = g' \in \mathrm{Arr}(\mathcal{E})(C,D).$$

Soit un préfoncteur $F : \mathcal{C}_0 \to \mathcal{E}$. Alors il existe un unique préfoncteur

$$G : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{E}$$

tel que

- $G \circ J \equiv F$;
- $G(g \circ f) \equiv G(g) \circ G(f)$;
- $G(\mathsf{id}_A) \equiv id_{G(A)}$;
- $G(\emptyset) \equiv \emptyset^{\mathcal{E}}$;
- $G(\mathsf{j}_A) \equiv j^{\mathcal{E}}_{G(A)}$;
- $G(\mathsf{push}\,(A,B,C,f,g)) \equiv push^{\mathcal{E}}(G(A),G(B),G(C),G(f),G(g))$;
- $G(\&_1(A,B,C,f,g)) \equiv \&_1^{\mathcal{E}}(G(A),G(B),G(C),G(f),G(g))$;
- $G(\&_2(A,B,C,f,g)) \equiv \&_2^{\mathcal{E}}(G(A),G(B),G(C),G(f),G(g))$;
- $G(\mathsf{up}\,(A,B,C,D,f,g,f',g') \equiv up^{\mathcal{E}}(G(A),G(B),G(C),G(D),G(f),G(g),G(f'),G(g'))$.

Nous commençons par montrer l'existence du préfoncteur

$$G : \mathrm{TERME}(\mathcal{C}_0) \to \mathcal{E}.$$

Pour cela, nous définissons pour tout entier i un préfoncteur $G_i : \mathcal{C}_i \to \mathcal{E}$, par induction sur i. Nous posons $G_0 \equiv F$, et définissons le préfoncteur $G_{i+1} : \mathcal{C}_{i+1} \to \mathcal{E}$ à partir du préfoncteur $G_i : \mathcal{C}_i \to \mathcal{E}$.

Hypothèses d'induction

Pour $i \in \mathbb{N}$, on fait les hypothèses d'induction suivantes.

1. Si $i \geq 1$,
   (a) $\forall A \in \mathrm{Obj}(\mathcal{C}_{i-1})$, $G_i(A) \equiv G_{i-1}(A)$;
   (b) $\forall A,B \in \mathrm{Obj}(\mathcal{C}_{i-1})$, $\forall f \in \mathrm{Arr}(\mathcal{C}_{i-1})(A,B)$, $G_i(f) \equiv G_{i-1}(f)$;
2. $h = h' \in \mathrm{Arr}(\mathcal{C}_i)(A,B) \Rightarrow G_i(h) = G_i(h') \in \mathrm{Arr}(\mathcal{E})(G_i(A),G_i(B))$;
3. $G_i : \mathcal{C}_i \to \mathcal{E}$ est un préfoncteur.

Pour $i = 0$, on pose $G_0 \equiv F : \mathcal{C}_0 \to \mathcal{E}$. Les hypothèses d'induction sont bien vérifiées.

Nous définissons maintenant le préfoncteur $G_{i+1} : \mathcal{C}_{i+1} \to \mathcal{E}$, en supposant que les hypothèses d'induction sont satisfaites au rang i.

Définition du préfoncteur $G_{i+1} : \mathcal{C}_{i+1} \to \mathcal{E}$

Le préfoncteur G_{i+1} est défini à partir de G_i, par induction sur la structure des objets et des flèches de $\mathcal{C}_{i+1}$.

- Action sur les objets :

  Règle (1) $G_{i+1}(A) \equiv G_0(A)$, pour $A \in \mathrm{Obj}(\mathcal{C}_0)$;

  Règle (2) $G_{i+1}(\emptyset) \equiv \emptyset^{\mathcal{E}}$;

  Règle (3) $G_{i+1}(\mathsf{push}\,(A, B, C, f, g))$
  $\equiv push^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g))$.

- Action sur les flèches :

  Règle (4) $G_{i+1}(f) \equiv G_0(f)$, pour $f \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$;

  Règle (5) $G_{i+1}(g \circ f) \equiv G_{i+1}(g) \circ G_{i+1}(f)$;

  Règle (6) $G_{i+1}(\mathsf{id}_A) \equiv id_{G_{i+1}(A)}$, identité de $\mathcal{E}$;

  Règle (7) $G_{i+1}(\mathsf{j}_A) \equiv j^{\mathcal{E}}_{G_{i+1}(A)}$;

  Règle (8) $G_{i+1}(\&_1(A, B, C, f, g)) \equiv \&_1^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g))$;

  Règle (9) $G_{i+1}(\&_2(A, B, C, f, g)) \equiv \&_2^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g))$;

  Règle (10) $G_{i+1}(\mathsf{up}\,(A, B, C, D, f, g, f', g'))$
  $\equiv up^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(D), G_i(f), G_i(g), G_i(f'), G_i(g'))$.
  Cette flèche existe car on a $f' \circ f = g' \circ g \in \mathrm{Arr}(\mathcal{C}_i)(A, D)$. Par hypothèse d'induction 3, G_i est un préfoncteur, donc

  $$G_i(f') \circ G_i(f) = G_i(g') \circ G_i(g) \in \mathrm{Arr}(\mathcal{E})(G_i(A), G_i(D)).$$

Pas d'induction

Nous montrons maintenant que les hypothèses d'induction sont bien vérifiées au rang $i + 1$, c'est-à-dire que

1. (a) $\forall A \in \mathrm{Obj}(\mathcal{C}_i)$, $G_{i+1}(A) \equiv G_i(A)$;
   (b) $\forall A, B \in \mathrm{Obj}(\mathcal{C}_i)$, $\forall f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$, $G_{i+1}(f) \equiv G_i(f)$;
2. $h = h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B) \Rightarrow G_{i+1}(h) = G_{i+1}(h') \in \mathrm{Arr}(\mathcal{E})(G_{i+1}(A), G_{i+1}(B))$;
3. $G_{i+1} : \mathcal{C}_{i+1} \to \mathcal{E}$ est un préfoncteur.

Preuve.

Point 1. On montre le point 1 (a) par induction sur la structure des objets de $\mathcal{C}_{i+1}$, le point 1 (b) par induction sur la structure des flèches de $\mathcal{C}_{i+1}$.

Point 2. On montre le point 2 par induction sur la longueur de la preuve que $h = h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$.

Règle (11) Si $h = h' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)$, alors $G_0(h) = G_0(h')$, et donc, par définition de G_{i+1}, $G_{i+1}(h) = G_{i+1}(h')$.

Règle (12) Si $h \equiv h'$, alors on a évidemment $G_{i+1}(h) \equiv G_{i+1}(h')$, d'où le résultat.

Règle (13) Si $h' = h \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$, alors par hypothèse d'induction, $G_{i+1}(h') = G_{i+1}(h)$, d'où $G_{i+1}(h) = G_{i+1}(h')$.

Règle (14) Si $h = g \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$ et $g = h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$, alors, par hypothèse d'induction, on a $G_{i+1}(h) = G_{i+1}(g)$ et $G_{i+1}(g) = G_{i+1}(h')$, d'où $G_{i+1}(h) = G_{i+1}(h')$.

Règle (15) On suppose que $h \equiv g \circ f$, $h' \equiv g' \circ f'$, $f = f' \in \mathrm{Arr}(\mathcal{C}_{i+1})(A, X)$, et $g = g' \in \mathrm{Arr}(\mathcal{C}_{i+1})(X, B)$. Par hypothèse d'induction, on a $G_{i+1}(f) = G_{i+1}(f')$ et $G_{i+1}(g) = G_{i+1}(g')$. Donc $G_{i+1}(g) \circ G_{i+1}(f) = G_{i+1}(g') \circ G_{i+1}(f')$, et donc, par définition de G_{i+1}, $G_{i+1}(g \circ f) = G_{i+1}(g' \circ f')$, d'où $G_{i+1}(h) = G_{i+1}(h')$.

Règle (16) Si $h \equiv (k \circ g) \circ f$ et $h' \equiv k \circ (g \circ f)$, par définition de G_{i+1},

$$\begin{aligned} G_{i+1}(h) &\equiv G_{i+1}((k \circ g) \circ f) \\ &\equiv G_{i+1}(k \circ g) \circ G_{i+1}(f) \\ &\equiv (G_{i+1}(k) \circ G_{i+1}(g)) \circ G_{i+1}(f) \\ &\equiv G_{i+1}(k) \circ (G_{i+1}(g) \circ G_{i+1}(f)) \\ &\equiv G_{i+1}(k \circ (g \circ f)) \\ &\equiv G_{i+1}(h') \end{aligned}$$

Règle (17) Si $h \equiv f \circ \mathsf{id}_A$, et $h' \equiv f$, on a

$$\begin{aligned} G_{i+1}(h) &\equiv G_{i+1}(f \circ \mathsf{id}_A) \\ &\equiv G_{i+1}(f) \circ G_{i+1}(\mathsf{id}_A) && \text{(définition de } G_{i+1}) \\ &\equiv G_{i+1}(f) \circ id_{G_{i+1}(A)} && \text{(définition de } G_{i+1}) \\ &= G_{i+1}(f) && \text{(identité dans } \mathcal{E}) \\ &\equiv G_{i+1}(h') \end{aligned}$$

Règle (18) Preuve similaire à celle de la règle (17).

Règle (19) Si $h, h' \in \mathrm{Arr}(\mathcal{C}_{i+1})(\emptyset, A)$, comme $\emptyset^{\mathcal{E}}$ est pré-initial, $G_{i+1}(h) = G_{i+1}(h')$.

Règle (20) $h \equiv \&_1(A, B, C, f, g) \circ f$ et $h' \equiv \&_2(A, B, C, f, g) \circ g$.

$$\begin{aligned} G_{i+1}(h) &\equiv G_{i+1}(\&_1(A, B, C, f, g) \circ f) \\ &\equiv G_{i+1}(\&_1(A, B, C, f, g)) \circ G_{i+1}(f) \\ &\equiv G_{i+1}(\&_1(A, B, C, f, g)) \circ G_i(f) && \text{(d'après 1 (b))} \\ &\equiv \&_1^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)) \circ G_i(f) \\ G_{i+1}(h') &\equiv G_{i+1}(\&_2(A, B, C, f, g) \circ g) \\ &\equiv G_{i+1}(\&_2(A, B, C, f, g)) \circ G_{i+1}(g) \\ &\equiv G_{i+1}(\&_2(A, B, C, f, g)) \circ G_i(g) && \text{(d'après 1 (b))} \\ &\equiv \&_2^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)) \circ G_i(g) \end{aligned}$$

Comme le triplet

$$\begin{array}{ll}(& push^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)),\\ & \&_1^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)),\\ & \&_2^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)) \ )\end{array}$$

est une pré-somme amalgamée dans $\mathcal{E}$, on a

$$\begin{aligned}&\&_1^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)) \circ G_i(f)\\ &\quad = \&_2^{\mathcal{E}}(G_i(A), G_i(B), G_i(C), G_i(f), G_i(g)) \circ G_i(g),\end{aligned}$$

et donc $G_{i+1}(h) = G_{i+1}(h')$.

Règle (21) $h \equiv \mathsf{up}(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g)$ et $h' \equiv f'$.

$$\begin{aligned}G_{i+1}(h) &\equiv G_{i+1}(\mathsf{up}(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g))\\ &\equiv G_{i+1}(\mathsf{up}(A, B, C, D, f, g, f', g')) \circ G_{i+1}(\&_1(A, B, C, f, g))\\ &= G_i(f')\\ &\equiv G_{i+1}(f')\end{aligned}$$

d'après la définition de $G_{i+1}(\mathsf{up}(A, B, C, D, f, g, f', g'))$.

Règle (22) Preuve similaire à celle de la règle (21).

Règle (23) $h \equiv u$ et $h \equiv v$, avec

$$\begin{aligned}u \circ \&_1(A, B, C, f, g) = v \circ \&_1(A, B, C, f, g) &\in \mathrm{Arr}(\mathcal{C}_{i+1})(B, D)\\ u \circ \&_2(A, B, C, f, g) = v \circ \&_2(A, B, C, f, g) &\in \mathrm{Arr}(\mathcal{C}_{i+1})(C, D).\end{aligned}$$

Par hypothèse d'induction, on a

$$\begin{aligned}G_{i+1}(u \circ \&_1(A, B, C, f, g)) &= G_{i+1}(v \circ \&_1(A, B, C, f, g))\\ G_{i+1}(u \circ \&_2(A, B, C, f, g)) &= G_{i+1}(v \circ \&_2(A, B, C, f, g))\end{aligned}$$

et donc

$$\begin{aligned}G_{i+1}(u) \circ G_{i+1}(\&_1(A, B, C, f, g)) &= G_{i+1}(v) \circ G_{i+1}(\&_2(A, B, C, f, g))\\ G_{i+1}(u) \circ G_{i+1}(\&_1(A, B, C, f, g)) &= G_{i+1}(v) \circ G_{i+1}(\&_2(A, B, C, f, g)).\end{aligned}$$

Par conséquent, par définition de $G_{i+1}(\mathsf{up}(A, B, C, D, f, g, f', g'))$, on a

$$G_{i+1}(u) = G_{i+1}(v).$$

Cela termine la preuve du point 2.

Point 3. On montre sans difficulté que $G_{i+1} : \mathcal{C}_{i+1} \to \mathcal{E}$ est bien un préfoncteur. □

Nous avons donc montré le résultat suivant.

Lemme A.1 *Pour tout entier i,*

1. (a) $\forall A \in \mathrm{Obj}(\mathcal{C}_i)$, $G_{i+1}(A) \equiv G_i(A)$;
   (b) $\forall A, B \in \mathrm{Obj}(\mathcal{C}_i)$, $\forall f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)$, $G_{i+1}(f) \equiv G_i(f)$;
2. *$G_i : \mathcal{C}_i \to \mathcal{E}$ est un préfoncteur.*

Définition du préfoncteur G : TERME($\mathcal{C}_0$) $\to \mathcal{E}$

On définit maintenant le préfoncteur G : TERME($\mathcal{C}_0$) $\to \mathcal{E}$ par son action sur les objets et les flèches de TERME($\mathcal{C}_0$).

- Action sur les objets.
  Pour tout $A \in \mathrm{Obj}(\mathrm{TERME}(\mathcal{C}_0))$, il existe $k \in \mathbb{N}$ tel que $A \in \mathrm{Obj}(\mathcal{C}_k)$. On pose
  $$G(A) \equiv G_k(A).$$
  D'après le lemme A.1 (a), cette définition est indépendante de l'entier k choisi.

- Action sur les flèches.
  Pour tout $f \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))$, il existe $k \in \mathbb{N}$ tel que $f \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$. On pose
  $$G(f) \equiv G_k(f).$$
  D'après le lemme A.1 (b), cette définition est indépendante de l'entier k choisi.

- Compatibilité avec les équivalences.
  Si $h = h' \in \mathrm{Arr}(\mathrm{TERME}(\mathcal{C}_0))(A, B)$, alors il existe $k \in \mathbb{N}$ tel que $h = h' \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$. Comme G_k est un préfoncteur, $G_k(h) = G_k(h')$. Par conséquent, $G(h) = G(h')$.

On vérifie que G est bien un préfoncteur tel que

- $G \circ J \equiv F$;
- $G(g \circ f) \equiv G(g) \circ G(f)$;
- $G(\mathsf{id}_A) \equiv id_{G(A)}$;
- $G(\emptyset) \equiv \emptyset^{\mathcal{E}}$;
- $G(\mathsf{j}_A) \equiv j^{\mathcal{E}}_{G(A)}$;
- $G(\mathsf{push}\,(A, B, C, f, g)) \equiv push^{\mathcal{E}}(G(A), G(B), G(C), G(f), G(g))$;
- $G(\&_1(A, B, C, f, g)) \equiv \&_1^{\mathcal{E}}(G(A), G(B), G(C), G(f), G(g))$;
- $G(\&_2(A, B, C, f, g)) \equiv \&_2^{\mathcal{E}}(G(A), G(B), G(C), G(f), G(g))$;
- $G(\mathsf{up}\,(A, B, C, D, f, g, f', g')$
  $\equiv up^{\mathcal{E}}(G(A), G(B), G(C), G(D), G(f), G(g), G(f'), G(g'))$.

Unicité du préfoncteur G

Soit un préfoncteur G' : TERME($\mathcal{C}_0$) $\to \mathcal{E}$ tel que

- $G' \circ J \equiv F$;
- $G'(g \circ f) \equiv G'(g) \circ G'(f)$;

- $G'(\mathsf{id}_A) \equiv id_{G'(A)}$;
- $G'(\emptyset) \equiv \emptyset^{\mathcal{E}}$;
- $G'(\mathsf{j}_A) \equiv j^{\mathcal{E}}_{G'(A)}$;
- $G'(\mathsf{push}\ (A, B, C, f, g)) \equiv push^{\mathcal{E}}(G'(A), G'(B), G'(C), G'(f), G'(g))$;
- $G'(\&_1(A, B, C, f, g)) \equiv \&_1^{\mathcal{E}}(G'(A), G'(B), G'(C), G'(f), G'(g))$;
- $G'(\&_2(A, B, C, f, g)) \equiv \&_2^{\mathcal{E}}(G'(A), G'(B), G'(C), G'(f), G'(g))$;
- $G'(\mathsf{up}\ (A, B, C, D, f, g, f', g')$
  $\equiv up^{\mathcal{E}}(G'(A), G'(B), G'(C), G'(D), G'(f), G'(g), G'(f'), G'(g'))$.

Par induction sur la structure des objets et flèches de $\mathrm{TERME}(\mathcal{C}_0)$, on a bien $G \equiv G'$.

Références bibliographiques

[AL91] A. Asperti and G. Longo. *Categories, Types and Structures, An Introduction to Category Theory for the Working Computer Scientist.* Foundations of Computing Science, MIT Press, 1991.

[AM75] M. A. Arbib and E. G. Manes. *Arrows, Structures and Functors: The Categorical Imperative.* Academic Press, 1975.

[B+90] D. Bert et al. Reference manual of the specification language LPG. Technical Report 59, LIFIA, Mars 1990. Anonymous ftp at `imag.fr`, in `/pub/SCOP/LPG/NewSun4/man_lpg.dvi`.

[BBC86] G. Bernot, M. Bidoit, and C. Choppy. Abstract data types with exception handling: An initial approach based on a distinction between exception and errors. *Theoretical Computer Science*, 46(1):13–45, 1986.

[BDMN73] G. Birtwistle, O.-J. Dahl, B. Myrhaug, and K. Nygaard. *Simula Begin.* Auerbach Pub., New York, 1973.

[BE86] D. Bert and R. Echahed. Design and implementation of a generic, logic and functional programming language. In *Proceedings of ESOP'86*, number 213 in LNCS, pages 119–132. Springer-Verlag, 1986.

[Ber83] D. Bert. Refinements of generic specifications with algebraic tools. In *Proceedings of IFIP'83, Paris*, pages 815–820, 1983.

[Ber90] D. Bert. Spécification de logiciels réutilisables. Technical Report RR-828-I-IMAG-116, LIFIA, Octobre 1990.

[BG77] R. M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1045–1058, 1977.

[BG80] R. M. Burstall and J. A. Goguen. The semantics of CLEAR, a specification language. In *Proceedings of Advanced Course on Abstract Software Specification*, number 86 in LNCS, pages 292–332. Springer-Verlag, 1980.

[BHK90] J. A. Bergstra, J. Heering, and R. Klint. Module algebra. *J. ACM*, 37(2):335–372, Apr. 1990.

[Bid89] M. Bidoit. PLUSS, un langage pour le développement de spécifications algébriques modulaires, Mai 1989. Thèse d'État, Université de Paris-Sud.

[BL93] G. Bernot and P. Le Gall. Exception handling and term labelling. In *Proceedings of TAPSOFT'93*, number 668 in LNCS, pages 421–436. Springer-Verlag, 1993.

[BR86] R. M. Burstall and D. Rydeheard. Computing with categories. Technical Report ECS-LFCS-86-9, University of Edinburgh, September 1986.

[Bur80] R. M. Burstall. Electronic category theory. In *Proceedings of the 9^{th} Symposium on Mathematical Foundations of Computer Science*, 1980.

[BW85] S. L. Bloom and E. G. Wagner. Many-sorted theories and their algebras with some applications to data types. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, chapter 4, pages 133–168. Cambridge University Press, 1985.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice-Hall International, 1990.

[BW94] M. Barr and C. Wells. The categorical theory generated by a limit sketch, Nov. 1994.

[Car86] J. Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[Coh65] P. M. Cohn. *Universal Algebra.* Harper and Row, 1965. Revised version 1980.

[Cur91] F. Cury. Catégories lax-localement-cartésiennes et catégories localement cartésiennes: un exemple de suffisante complétude connexe de sémantiques initiales. In *diagrammes*, volume 25, pages 1–155, Université Paris 7, Juillet 1991.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science*, chapter 15. Elsevier Science Publishers B.V., 1990.

[DN66] O.-J. Dahl and K. Nygaard. Simula – an Algol-based simulation language. *Communications of the ACM*, 9:671–678, Sept. 1966.

[DR94a] D. Duval and J.-C. Reynaud. Sketches and computation (part 1): Basic definitions and static evaluation. *Mathematical Structures in Computer Science*, 4:185–238, 1994.

[DR94b] D. Duval and J.-C. Reynaud. Sketches and computation (part 2): Dynamic evaluation and applications. *Mathematical Structures in Computer Science*, 4:239–271, 1994.

[Ehr65] C. Ehresmann. *Catégories et structures.* Dunod, Paris, 1965.

[Ehr68] C. Ehresmann. Esquisses et types de structures algébriques. *Bulletin de l'Institut Polytechnique*, Iasi 14, 1968.

[EJO93] H. Ehrig, R. M. Jimenez, and F. Orejas. Compositionality results for different types of parameterization and parameter passing in specification languages. In *Proceedings of the 4^{th} International Joint Conference CAAP/FASE*, number 668 in LNCS, pages 31–45. Springer-Verlag, 1993.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1. Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1985.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2. Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1990.

[FGJM85] K. Futatsugi, J. A. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proceedings of Principles of Programming Languages*, pages 52–66, 1985.

[Gau84] M.-C. Gaudel. A first introduction to PLUSS. Technical report, Université d'Orsay, France, 1984.

[GB84] J. A. Goguen and R. M. Burstall. Introducing institutions. In *Proceedings of the Workshop on Logic of Programming*, number 164 in LNCS, pages 221–256. Springer-Verlag, 1984.

[GB90] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. Research Report ECS-LFCS-90-106, University of Edinburgh, January 1990.

[GKK+87] J. A. Goguen, C. Kirchner, H. Kirchner, A. Mégrelis, J. Meseguer, and T. Winkler. An introduction to OBJ3. In *Proceedings of the 1^{st} International Workshop on Conditional Term Rewriting Systems*, number 308 in LNCS, pages 258–263. Springer-Verlag, 1987.

[Gog73] J. A. Goguen. Categorical foundations for general systems theory. In *Advances in Cybernetics and System Research*, pages 121–130. Transcripta Books, 1973.

[Gog92] J. A. Goguen. Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 2:159–191, 1992.

[Gol79] R. Goldblatt. *Topoi: The Categorial Analysis of Logic*, volume 98 of *Studies in Logic and the Foundations of Mathematics.* North Holland, 1979.

[GTW78] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In R. T. Yeh, editor, *Current Trends in Programming Methodology*, volume 4: Data Structuring, pages 80–149. Prentice-Hall, 1978.

[GTWW75] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Abstract data types as initial algebras and the correctness of data representation. In *Computer Graphics, Pattern Recognition and Data Structure*, pages 89–93, 1975.

[GTWW77] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial algebra semantics and continuous algebra. *J. ACM*, 24:68–95, 1977.

[Gut75] J. V. Guttag. *The specification and application to programming of abstract data types.* PhD thesis, University of Toronto, 1975.

[Gut77] J. V. Guttag. Abstract data types and the development of data structures. *Communication of the ACM*, 6:396–404, 1977.

[Huf92] J.-M. Hufflen. Proposal for GLIDER version 1.0: Principles and main features. ICARUS Technical Report, INRIA-LORRAINE & CRIN, 1992.

[Law63] W. Lawvere. Functorial semantics of algebraic theories. *Proc. Nat. Acad. Sci.*, 50:869–873, 1963.

[LCW85] D. Lorge Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. *IEEE Transactions on Software Engineering*, SE-11(3):259–266, March 1985.

[LS86] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic.* Cambridge studies in advanced mathematics, 1986.

[LZ74] B. Liskov and S. Zilles. Programming with abstract data types. *ACM SIGPLAN Notices*, 9(4):50–59, 1974.

[Mar95] A. Martins. *La généralisation: un outil pour la réutilisation.* PhD thesis, INPG, Grenoble, Mars 1995.

[MB70] S. Mac Lane and G. Birkhoff. *Algèbre. 1. Structures fondamentales.* Gauthier-Villars, 1970.

[McL71] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[MG85] J. Meseguer and J. A. Goguen. Initiality, induction, and computability. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, chapter 14, pages 459–541. Cambridge University Press, 1985.

[Ori94] C. Oriat. Representing modular specifications as diagrams. In *Compte Rendu des Journées du GDR Programmation, Lille*, pages 53–68, Septembre 1994.

[Ori95] C. Oriat. Detecting isomorphisms of modular specifications with diagrams. In *Proceedings of AMAST'95*, number 936 in LNCS, pages 184–198. Springer-Verlag, 1995.

[Poi92] A. Poigné. Basic category theory. In *Handbook of Logic in Computer Science, Volume 1. Background: Mathematical Structures*, pages 413–640. Oxford Science Publication, 1992.

[Ren91] G. Renardel de Lavalette. Logical semantics of modularisation. In *Proceedings of CSL '91*, number 626 in LNCS, pages 306–315. Springer-Verlag, 1991.

[Rey90a] J.-C. Reynaud. Putting algebraic components together: A dependent type approach. Research Report 810 I IMAG, LIFIA, Avril 1990. Extended version.

[Rey90b] J.-C. Reynaud. Putting algebraic components together: A dependent type approach. In *Proceedings of DISCO'90*, number 429 in LNCS. Springer-Verlag, 1990.

[Rey93] J.-C. Reynaud. Isomorphism of typed algebraic specifications. Internal Report, LGI-IMAG, Avril 1993.

[See84] R. A. G. Seely. Locally cartesian closed categories and type theory. *Math. Proc. Camb. Phil. Soc.*, 95(33):33–48, 1984.

[SP77] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. In *Proceedings of the 18^{th} IEEE Symposium on Foundations of Computer Science*, pages 13–17, 1977.

[SST92] D. Sannella, S. Sokolowski, and A. Tarlecki. Toward formal development of programs from algebraic specifications: Parameterisation revisited. *Acta Informatica*, 29:689–736, 1992.

[ST88] D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

[SW83] D. Sannella and M. Wirsing. A kernal language for algebraic specification and implementation. In *Proceedings of the 11^{th} Colloquium on Foundations of Computation Theory*, number 158 in LNCS, pages 413–427. Springer-Verlag, 1983.

[SW91] T. Streicher and M. Wirsing. Dependent types considered necessary for specification languages. In *Proceedings of the 7^{th} Workshop on Specification of Abstract Data Types*, number 534 in LNCS, pages 323–339, 1991.

[TBG91] A. Tarlecki, R. M. Burstall, and J. A. Goguen. Some fundamental algebraic tools for the semantics of computation: Part 3. Indexed categories. *Theoretical Computer Science*, 91:239–264, 1991.

[TWW82] J. W. Thatcher, E. G. Wagner, and J. B. Wright. Data type specification: Parameterization and the power of specification techniques. *ACM Trans. Prog. Lang. Syst.*, 4:711–773, 1982.

[Wan79] M. Wand. Fixed-point constructions in order-enriched categories. *Theoretical Computer Science*, 8:13–30, 1979.

[WBT85] E. G. Wagner, S. L. Bloom, and J. W. Thatcher. Why algebraic theories? In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, chapter 17, pages 607–634. Cambridge University Press, 1985.

[Wir86] M. Wirsing. Structured algebraic specifications: A kernel language. *Theoretical Computer Science*, 42:123–249, 1986.

[Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 13, pages 677–788. Elsevier Science Publishers B.V., 1990.

[Wir94] M. Wirsing. Algebraic specification languages: An overview. In *Recent Trends in Data Type Specification, 10^{th} Workshop on Specification of Abstract Data Types*, number 906 in LNCS, 1994.

Table de la deuxième partie

Laboratoire LSR (Logiciels, Systèmes, Réseaux)
Institut IMAG
(Institut d'Informatique et de Mathématiques Appliquées de Grenoble)
681, rue de la Passerelle
B.P. 72
38402 Saint Martin d'Hères Cedex