DIAGRAMMES

C. ORIAT

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

Diagrammes, tome 41 (1999), p. 3-76

<http://www.numdam.org/item?id=DIA_1999__41__3_0>

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

PARTIE III (**)

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 II dans Diagrammes 40.

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 les deux premières parties, comprenant les chapitres 1, 2 et 3, nous avons présenté le cadre général de notre travail (les spécifications modulaires), les bases théoriques (les catégories de diagrammes) et la syntaxe des spécifications modulaires.

Ce volume contient la troisième partie de notre thèse, c'est-à-dire les chapitres 4 et 5, consacrés respectivement à la traduction des spécifications modulaires en diagrammes et à la détection d'isomorphismes entre diagrammes.

Dans le chapitre 4, nous interprétons les termes dénotant des spécifications modulaires par des diagrammes. Cette interprétation permet d'obtenir une représentation plus abstraite des spécifications modulaires que celle donnée par les termes, parce que certaines étapes spécifiques de la construction sont éliminées. Nous montrons que l'interprétation des termes par les diagrammes définit une équivalence entre les catégories Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$). Ce résultat nous permet de déduire que toute spécification modulaire est isomorphe à la colimite du diagramme qui la représente.

Le chapitre 5 est une application de cette représentation des spécifications modulaires par des diagrammes. Nous proposons de détecter un isomorphisme de construction entre deux spécifications modulaires en détectant un isomorphisme entre leurs diagrammes correspondants. Nous présentons un algorithme qui permet, dans l'hypothèse où la catégorie de base $\mathcal{C}_0$ est finie et ne comporte pas de cycle, de détecter si deux diagrammes sont isomorphes.

Chapitre 4

Sémantique : des termes aux diagrammes

Le but de ce chapitre est d'associer à toute spécification modulaire un *diagramme*, et à tout morphisme de spécifications modulaires un *morphisme de diagrammes*. On appelle $\mathcal{C}_0$ la catégorie des spécifications et morphismes de spécifications de base, et on décrit cette association par un préfoncteur

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

Intuitivement, étant donné un objet A de TERME($\mathcal{C}_0$) qui représente une spécification modulaire, $\mathcal{D}(A)$ est un diagramme dont la colimite est isomorphe à cette spécification. Le foncteur $\mathcal{D}$ décrit une *interprétation* des termes par des diagrammes, il s'agit donc d'une *sémantique* pour les spécifications algébriques modulaires, par opposition aux termes qui constituent la *syntaxe*.

Après avoir défini le préfoncteur $\mathcal{D}$: TERME($\mathcal{C}_0$) $\to$ DIAGR($\mathcal{C}_0$), nous détaillons les calculs de diagrammes associés aux spécifications modulaires A_2, A_2', A_3 et A_4 des anneaux présentées dans le chapitre 1 (partie I).

Dans la dernière partie, nous montrons que les précatégories TERME($\mathcal{C}_0$) et DIAGR($\mathcal{C}_0$) sont équivalentes, ce qui implique par passage au quotient que les catégories associées Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$) sont équivalentes. Intuitivement, cela signifie que ces deux catégories ont "la même structure", "aux isomorphismes entre objets près". Nous en déduisons que deux objets de Terme($\mathcal{C}_0$) sont isomorphes si et seulement si leurs diagrammes associés sont isomorphes dans la catégorie diagr($\mathcal{C}_0$). Enfin, nous montrons que le calcul de diagrammes est correct, c'est-à-dire qu'une spécification est isomorphe à la colimite du diagramme qui la représente.

4.1 Sémantique

4.1.1 Précatégorie DIAGR($\mathcal{C}_0$)

Dans tout ce chapitre, $\mathcal{C}_0$ est une petite catégorie. Rappelons que la catégorie DIAGR($\mathcal{C}_0$) peut être considérée comme une *précatégorie* de la façon suivante (cf.

chapitre 3, partie II, page 19) :

- l'identité syntaxique dans la précatégorie DIAGR($\mathcal{C}_0$) est l'égalité dans la catégorie DIAGR($\mathcal{C}_0$) ;
- l'équivalence dans la précatégorie DIAGR($\mathcal{C}_0$) est définie par la congruence $\approx$ sur les flèches de la catégorie DIAGR($\mathcal{C}_0$).

Par conséquent, la catégorie associée à la précatégorie DIAGR($\mathcal{C}_0$) est diagr($\mathcal{C}_0$).

La précatégorie DIAGR($\mathcal{C}_0$) est finiment pré-cocomplète. Plus précisément, la précatégorie DIAGR($\mathcal{C}_0$) a un objet pré-initial choisi et des pré-sommes amalgamées choisies.

Objet pré-initial

Rappelons que le diagramme vide, noté $\bigcirc$, est le diagramme dont le graphe sous-jacent ne contient aucun nœud et aucun arc.

Lemme 4.1 *Le diagramme vide est pré-initial dans la précatégorie* DIAGR($\mathcal{C}_0$). *Autrement dit,*

1. *pour tout diagramme* $\overline{\alpha}$ *de* DIAGR($\mathcal{C}_0$), *il existe une flèche*

$$\overline{j}_{\overline{\alpha}} : \bigcirc \to \overline{\alpha}$$

   dans DIAGR($\mathcal{C}_0$) ;

2. *pour toute flèche* $\overline{\sigma} : \bigcirc \to \overline{\alpha}$, *on a* $\overline{j}_{\overline{\alpha}} \approx \overline{\sigma}$.

Remarquons qu'en réalité le diagramme vide est *initial* dans DIAGR($\mathcal{C}_0$). En effet, pour toute flèche $\overline{\sigma} : \bigcirc \to \overline{\alpha}$, on a $\overline{j}_{\overline{\alpha}} = \overline{\sigma}$.

Pré-sommes amalgamées

Soit trois diagrammes $\overline{\alpha}$, $\overline{\beta}$ et $\overline{\gamma}$, et deux morphismes de diagrammes $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ et $\overline{\tau} : \overline{\alpha} \to \overline{\gamma}$ de DIAGR($\mathcal{C}_0$). On considère le diagramme de somme amalgamée

$$\overline{\overline{\Pi}} = (\Pi^{\Phi},\ \Pi : \Pi^{\Phi} \to \text{DIAGR}(\mathcal{C}_0))$$

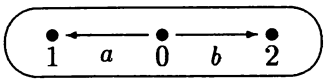
où

$$\begin{aligned} \Pi(0) &= \overline{\alpha}; \\ \Pi(1) &= \overline{\beta}; \\ \Pi(2) &= \overline{\gamma}; \\ \Pi(a) &= \overline{\sigma}; \\ \Pi(b) &= \overline{\tau}. \end{aligned}$$

Ce diagramme sera noté

$$\overline{\overline{\Pi}} = \mathcal{PushDiagr}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}).$$

Le diagramme $\overline{\overline{\Pi}}$ est un objet de $\mathrm{DIAGR}^2(\mathcal{C}_0)$, construit sur le graphe Π^Φ.



graphe Π^Φ

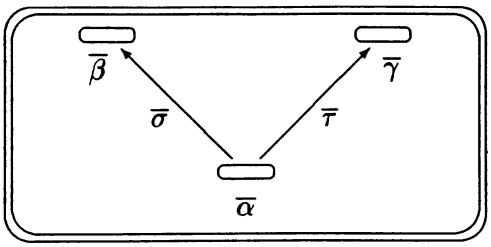


diagramme $\overline{\overline{\Pi}} = \mathcal{PushDiagr}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau})$

En aplatissant le diagramme $\overline{\overline{\Pi}}$, on obtient un diagramme $\overline{\delta}$ de $\mathrm{DIAGR}(\mathcal{C}_0)$ (cf. définition 2.29, partie I, page 84) :

$$\overline{\delta} = \mathrm{Apl}_{\mathcal{C}_0}(\overline{\overline{\Pi}}).$$

On a également deux flèches $\overline{J_1} : \overline{\beta} \to \overline{\delta}$ et $\overline{J_2} : \overline{\gamma} \to \overline{\delta}$ dans $\mathrm{DIAGR}(\mathcal{C}_0)$ (cf. définition 2.30, partie I, page 85) qui seront notées

$$\overline{J_1} = \overline{J_1}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) : \overline{\beta} \to \overline{\delta}$$
$$\overline{J_2} = \overline{J_2}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) : \overline{\gamma} \to \overline{\delta}.$$

Lemme 4.2 *Le triplet*

$$(\mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau})),\ \overline{J_1}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}),\ \overline{J_2}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}))$$

est une pré-somme amalgamée de $\overline{\beta}$ et $\overline{\gamma}$ par rapport aux flèches $\overline{\sigma}$ et $\overline{\tau}$. Cela signifie qu'on a les propriétés suivantes.

1. $\overline{J_1}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \circ \overline{\sigma} \approx \overline{J_2}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \circ \overline{\tau}$.
2. *Soit un diagramme $\overline{\delta'}$ et deux flèches*

$$\overline{K_1} : \overline{\beta} \to \overline{\delta'}$$
$$\overline{K_2} : \overline{\gamma} \to \overline{\delta'}$$

de $\mathrm{DIAGR}(\mathcal{C}_0)$ *telles que*

$$\overline{K_1} \circ \overline{\sigma} \approx \overline{K_2} \circ \overline{\tau}.$$

Alors,

(a) *en posant $\overline{\delta} = \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau})$, il existe une flèche*

$$\overline{U}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{K_1}, \overline{K_2}) : \overline{\delta} \to \overline{\delta'}$$

de $\mathrm{DIAGR}(\mathcal{C}_0)$ *telle que*

$$\begin{cases} \overline{U}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{K_1}, \overline{K_2}) \circ \overline{J_1}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \approx \overline{K_1} \\ \overline{U}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{K_1}, \overline{K_2}) \circ \overline{J_2}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \approx \overline{K_2} \ ; \end{cases}$$

(b) *pour toute flèche* $\overline{U'} : \overline{\delta} \to \overline{\delta'}$ *de* DIAGR($\mathcal{C}_0$) *telle que*

$$\begin{cases} \overline{U'} \circ \overline{J_1}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \approx \overline{K_1} \\ \overline{U'} \circ \overline{J_2}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \approx \overline{K_2} \end{cases}$$

on a

$$\overline{U}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{K_1}, \overline{K_2}) \approx \overline{U'}.$$

Preuve.

1. Il s'agit d'une reformulation du lemme 2.17, partie I, page 100.
2. Il s'agit d'une reformulation du lemme 2.19, partie I, page 101. Remarquons que la preuve de ce lemme est constructive, c'est-à-dire construit explicitement un choix de flèche

$$\overline{U}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{K_1}, \overline{K_2}) : \overline{\delta} \to \overline{\delta'}.$$

□

4.1.2 Préfoncteur $\mathcal{D}$: TERME($\mathcal{C}_0$) → DIAGR($\mathcal{C}_0$)

Rappelons qu'on a un préfoncteur $I_{\mathcal{C}_0} : \mathcal{C}_0 \to$ DIAGR($\mathcal{C}_0$). D'après les lemmes 4.1 et 4.2, on peut appliquer le théorème 3.9, partie II, page 36. Il existe donc un unique préfoncteur

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

tel que

1. $\forall A \in \text{Obj}(\mathcal{C}_0),\ \mathcal{D}(A) \equiv I_{\mathcal{C}_0}(A)$;
2. $\mathcal{D}(\emptyset) \equiv \bigcirc$;
3. $\mathcal{D}(\textsf{push}\ (A, B, C, f, g)) \equiv$
   $\text{Apl}_{\mathcal{C}_0}(\mathcal{P}ush\mathcal{D}iagr(\mathcal{D}(A), \mathcal{D}(B), \mathcal{D}(C), \mathcal{D}(f), \mathcal{D}(g)))$;
4. $\forall f \in \text{Arr}(\mathcal{C}_0)(A, B),\ \mathcal{D}(f) \equiv I_{\mathcal{C}_0}(f)$;
5. $\mathcal{D}(g \circ f) \equiv \mathcal{D}(g) \circ \mathcal{D}(f)$;
6. $\mathcal{D}(\textsf{id}_A) \equiv \overline{id}_{\mathcal{D}(A)}$;
7. $\mathcal{D}(\textsf{j}_A) \equiv \overline{j}_{\mathcal{D}(A)}$;
8. $\mathcal{D}(\&_1(A, B, C, f, g)) \equiv \overline{J_1}(\mathcal{D}(A), \mathcal{D}(B), \mathcal{D}(C), \mathcal{D}(f), \mathcal{D}(g))$;
9. $\mathcal{D}(\&_2(A, B, C, f, g)) \equiv \overline{J_2}(\mathcal{D}(A), \mathcal{D}(B), \mathcal{D}(C), \mathcal{D}(f), \mathcal{D}(g))$;
10. $\mathcal{D}(\textsf{up}\ (A, B, C, D, f, g, f', g'))$
    $\equiv \overline{U}(\mathcal{D}(A), \mathcal{D}(B), \mathcal{D}(C), \mathcal{D}(D), \mathcal{D}(f), \mathcal{D}(g), \mathcal{D}(f'), \mathcal{D}(g'))$.

Ces règles donnent une procédure de calcul du diagramme ou morphisme de diagrammes associé à un terme. Remarquons que nous avons remplacé la condition $\mathcal{D} \circ J \equiv I_{\mathcal{C}_0}$ du théorème 3.9 par les points 1 et 4, qui sont évidemment équivalents.

Enfin, au préfoncteur $\mathcal{D}$: TERME($\mathcal{C}_0$) $\rightarrow$ DIAGR($\mathcal{C}_0$) est évidemment associé à un *foncteur*

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

4.2 Exemple des anneaux

Dans cette section, nous calculons les diagrammes associés aux spécifications modulaires des anneaux A_2, A_2', A_3 et A_4 présentées dans le chapitre 1. Pour chaque spécification modulaire A, nous calculons donc $\mathcal{D}(A)$, en utilisant les règles de calcul du préfoncteur

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

décrites dans le paragraphe 4.1.2.

4.2.1 Diagramme associé à la spécification A_2

Dans le chapitre 1, nous avons défini une spécification modulaire des anneaux A_2 de la façon suivante (cf. partie I, page 41). Nous commençons par construire une spécification MD en combinant la spécification M des monoïdes avec la spécification D qui contient deux opérateurs distributifs, et en partageant la spécification de l'opérateur binaire multiplicatif. Puis on obtient une spécification des anneaux en combinant MD avec la spécification G des groupes, et en partageant l'opérateur additif.

$$MD \equiv \textsf{push}\ (B, M, D, b, m_*)$$
$$A_2 \equiv \textsf{push}\ (B, MD, G, \&_2(MD) \circ m_+, m \circ b)$$

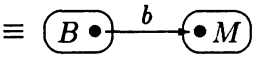
Nous calculons d'abord le diagramme associé à MD.

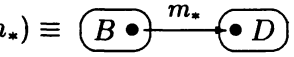
$$\begin{aligned} \mathcal{D}(MD) &\equiv \mathcal{D}(\textsf{push}\ (B, M, D, b, m_*)) \\ &\equiv \text{Apl}_{\mathcal{C}_0}(\mathcal{P}ush\mathcal{D}iagr(\mathcal{D}(B), \mathcal{D}(M), \mathcal{D}(D), \mathcal{D}(b), \mathcal{D}(m_*))) \end{aligned}$$

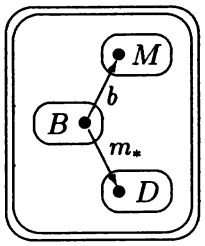
$\mathcal{D}(B) \equiv I_{\mathcal{C}_0}(B) \equiv$ 

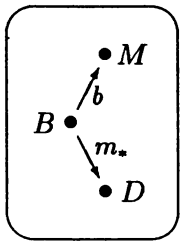
$\mathcal{D}(M) \equiv I_{\mathcal{C}_0}(M) \equiv$ 

$\mathcal{D}(D) \equiv I_{\mathcal{C}_0}(D) \equiv$ 

$\mathcal{D}(b) \equiv I_{\mathcal{C}_0}(b) \equiv$ 

$\mathcal{D}(m_*) \equiv I_{\mathcal{C}_0}(m_*) \equiv$ 

$\mathcal{D}(MD) \equiv \mathrm{Apl}_{\mathcal{C}_0}$ 

$\equiv$ 

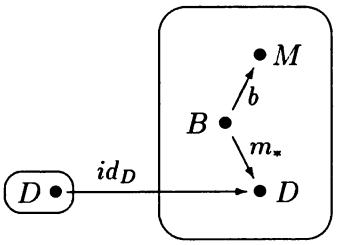
Nous calculons maintenant le diagramme associé à A_2.

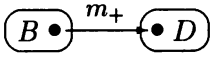
$$\begin{aligned}\mathcal{D}(A_2) &\equiv \mathcal{D}(\textsf{push}\,(B, MD, G, \&_2(MD) \circ m_+, m \circ b)) \\ &\equiv \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{P}ushDiagr(\mathcal{D}(B), \mathcal{D}(MD), \mathcal{D}(G), \mathcal{D}(\&_2(MD) \circ m_+), \mathcal{D}(m \circ b)))\end{aligned}$$

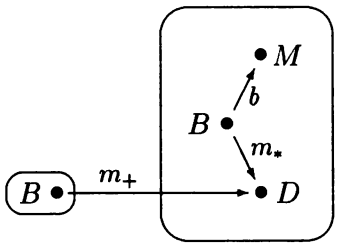
$\mathcal{D}(G) \equiv I_{\mathcal{C}_0}(G) \equiv$ 

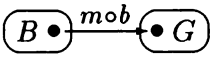
$\mathcal{D}(\&_2(MD) \circ m_+) \equiv \mathcal{D}(\&_2(MD)) \circ \mathcal{D}(m_+)$

$\mathcal{D}(\&_2(MD)) \equiv \overline{J_2}(\mathcal{D}(B), \mathcal{D}(M), \mathcal{D}(D), \mathcal{D}(b), \mathcal{D}(m_*))$

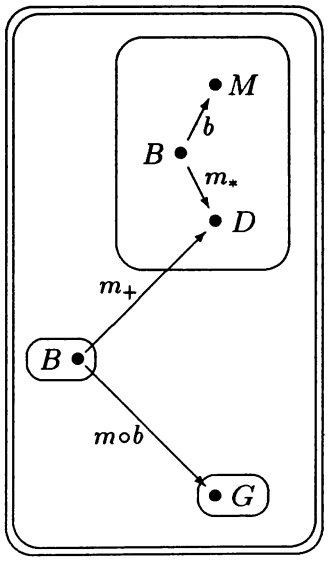
$\equiv$ 

$\mathcal{D}(m_+) \equiv$ 

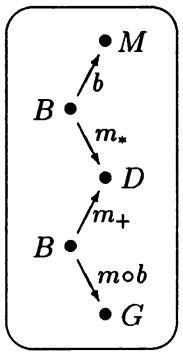
$\mathcal{D}(\&_2(MD) \circ m_+) \equiv$ 

$\mathcal{D}(m \circ b) \equiv$ 

$\mathcal{D}(A_2) \equiv \mathrm{Apl}_{\mathcal{C}_0}$



$\equiv$



$\equiv \overline{\delta_2}$

Finalement, le diagramme associé à la spécification A_2 est le diagramme $\overline{\delta_2}$ que nous avions donné partie I, page 43.

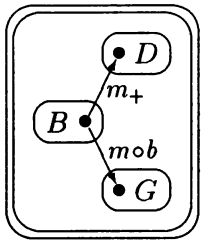
4.2.2 Diagramme associé à la spécification A'_2

Nous calculons maintenant le diagramme associé à la spécification A'_2 des anneaux. La spécification A'_2 a été construite en commençant par combiner les spécifications D et G, puis en ajoutant M (cf. partie I, page 42).

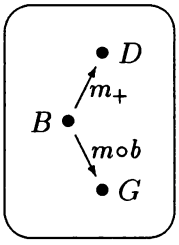
$DG \equiv \mathbf{push}\,(B, D, G, m_+, m \circ b)$
$A'_2 \equiv \mathbf{push}\,(B, M, DG, b, \&_1(DG) \circ m_*)$

$\mathcal{D}(DG) \equiv \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\mathcal{D}(B), \mathcal{D}(D), \mathcal{D}(G), \mathcal{D}(m_+), \mathcal{D}(m) \circ \mathcal{D}(b)))$

$\equiv \mathrm{Apl}_{\mathcal{C}_0}$

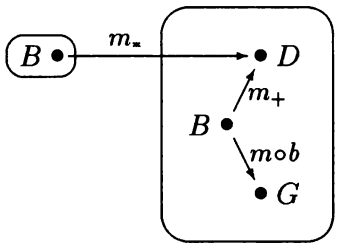


$\equiv$

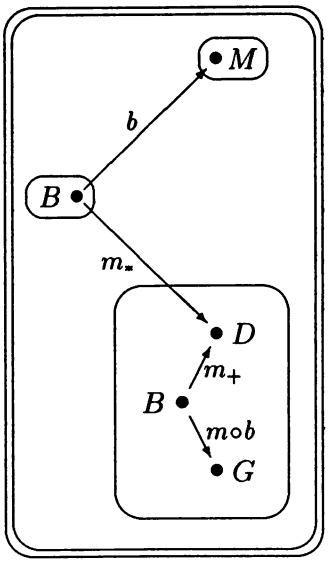


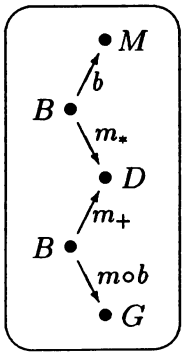
$\mathcal{D}(\&_1(DG) \circ m_*) \equiv \mathcal{D}(\&_1(DG)) \circ \mathcal{D}(m_*))$
$\equiv \overline{J_1}(\mathcal{D}(B), \mathcal{D}(D), \mathcal{D}(G), \mathcal{D}(m_+), \mathcal{D}(m) \circ \mathcal{D}(b)) \circ \mathcal{D}(m_*)$

$\equiv$



$\mathcal{D}(A'_2) \equiv \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\mathcal{D}(B), \mathcal{D}(M), \mathcal{D}(DG), \mathcal{D}(\&_1(DG)) \circ \mathcal{D}(m_*)))$

$\equiv \mathrm{Apl}_{\mathcal{C}_0}$ 

$\equiv$ 

$\equiv \overline{\delta_2}$

Le diagramme associé à la spécification A'_2 est donc, comme pour la spécification A_2, le diagramme $\overline{\delta_2}$.

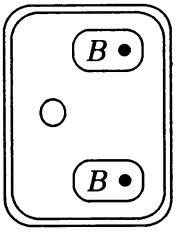
4.2.3 Diagramme associé à la spécification A_3

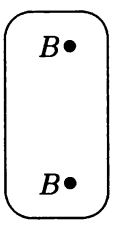
Pour la spécification A_3 des anneaux, nous avons utilisé deux spécifications intermédiaires : la spécification B_2 qui regroupe deux opérateurs binaires, et la spécification P des pseudo-anneaux (cf. partie I, page 43). Rappelons qu'un pseudo-anneau est un anneau sans la propriété de distributivité entre les deux opérateurs.

$B_2 \equiv \mathsf{push}\,(\emptyset, B, B, \mathrm{j}_B, \mathrm{j}_B)$
$u_2 \equiv \mathsf{up}\,(\emptyset, B, B, D, \mathrm{j}_B, \mathrm{j}_B, m_*, m_+) : B_2 \to D$
$P \equiv \mathsf{push}\,(S, M, G, b \circ s, m \circ b \circ s)$
$u_1 \equiv \mathsf{up}\,(\emptyset, B, B, P, \mathrm{j}_B, \mathrm{j}_B, \&_1(P) \circ b, \&_2(P) \circ m \circ b) : B_2 \to P$
$A_3 \equiv \mathsf{push}\,(B_2, P, D, u_1, u_2)$

Commençons par calculer le diagramme associé à la spécification B_2.
$\mathcal{D}(B_2) \equiv \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(\mathrm{j}_B), \mathcal{D}(\mathrm{j}_B)))$
$\mathcal{D}(\emptyset) \equiv \bigcirc$
$\mathcal{D}(B) \equiv$

$\mathcal{D}(\mathrm{j}_B) \equiv \bigcirc \to$ 

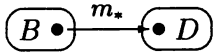
$\mathcal{D}(B_2) \equiv \mathrm{Apl}_{\mathcal{C}_0}$ 

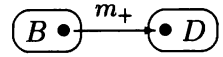
$\equiv$ 

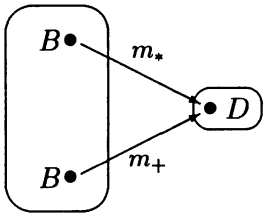
Calculons le morphisme de diagrammes associé au morphisme de spécifications u_2 : $B_2 \to D$.

$\mathcal{D}(u_2) \equiv \mathcal{D}(\mathsf{up}(\emptyset, B, B, D, \mathsf{j}_B, \mathsf{j}_B, m_*, m_+))$
$\equiv \overline{U}(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(D), \mathcal{D}(\mathsf{j}_B), \mathcal{D}(\mathsf{j}_B), \mathcal{D}(m_*), \mathcal{D}(m_+))$

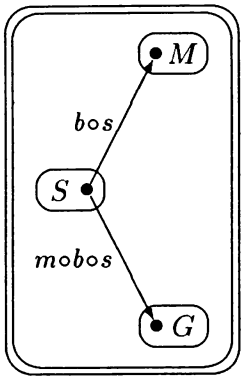
$\mathcal{D}(D) \equiv$ 

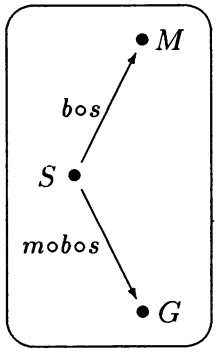
$\mathcal{D}(m_*) \equiv$ 

$\mathcal{D}(m_+) \equiv$ 

$\mathcal{D}(u_2) \equiv$ 

Calculons le diagramme associé à la spécification P des pseudo-anneaux.

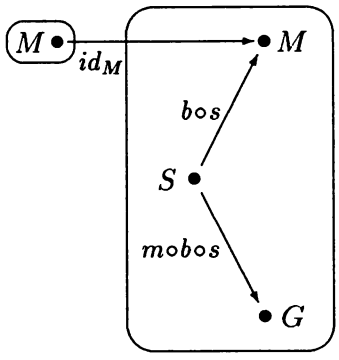
$\mathcal{D}(P) \equiv \mathrm{Apl}_{\mathcal{C}_0}$ 

$\equiv$ 

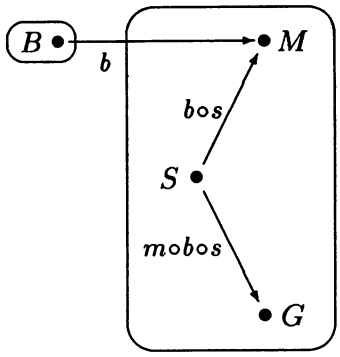
$\equiv \overline{\alpha}$ (cf. partie I, page 44)

Pour calculer le morphisme de diagrammes associé au morphisme de spécifications $u_1 : B_2 \to P$, nous devons d'abord calculer $\mathcal{D}(\&_1(P) \circ b)$ et $\mathcal{D}(\&_2(P) \circ m \circ b)$.

$\mathcal{D}(\&_1(P)) \equiv \overline{J_1}(\mathcal{D}(S), \mathcal{D}(M), \mathcal{D}(G), \mathcal{D}(b \circ s), \mathcal{D}(m \circ b \circ s))$

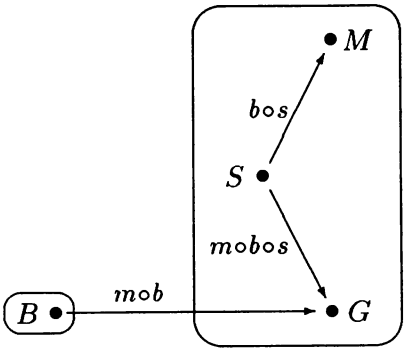
$\equiv$ 

$\mathcal{D}(\&_1(P) \circ b) \equiv$



$\mathcal{D}(\&_2(P) \circ m \circ b) \equiv \overline{J_2}(\mathcal{D}(S), \mathcal{D}(M), \mathcal{D}(G), \mathcal{D}(b \circ s), \mathcal{D}(m \circ b \circ s)) \circ \mathcal{D}(m) \circ \mathcal{D}(b)$

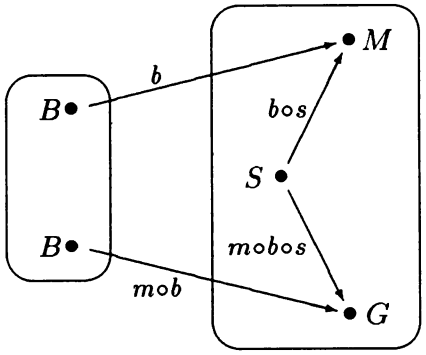
$\equiv$



Nous calculons maintenant le morphisme de diagrammes associé à $u_1 : B_2 \to P$.

$$\begin{aligned}\mathcal{D}(u_1) &\equiv \mathcal{D}(\mathsf{up}\,(\emptyset, B, B, P, \mathsf{j}_B, \mathsf{j}_B, \&_1(P) \circ b, \&_2(P) \circ m \circ b)) \\ &\equiv \overline{U}(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(P), \\ &\qquad \mathcal{D}(\mathsf{j}_B), \mathcal{D}(\mathsf{j}_B), \mathcal{D}(\&_1(P) \circ b), \mathcal{D}(\&_2(P) \circ m \circ b))\end{aligned}$$

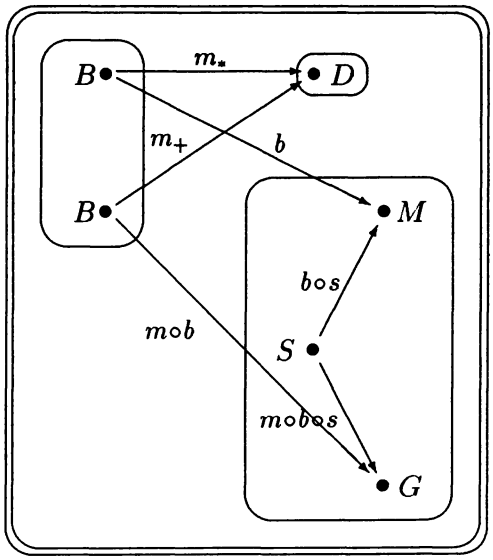
$\equiv$



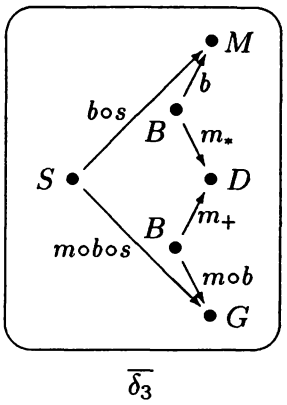
Nous calculons enfin le diagramme associé à la spécification A_3.

$$\begin{aligned}\mathcal{D}(A_3) &\equiv \mathcal{D}(\mathsf{push}\,(B_2, P, D, u_1, u_2))\\ &\equiv \mathrm{Apl}_{\mathcal{C}_0}(\mathcal{PushDiagr}(\mathcal{D}(B_2), \mathcal{D}(P), \mathcal{D}(D), \mathcal{D}(u_1), \mathcal{D}(u_2)))\end{aligned}$$

$\equiv \mathrm{Apl}_{\mathcal{C}_0}$



$\equiv$



$\overline{\delta_3}$

Le diagramme associé à la spécification A_3 est donc le diagramme $\overline{\delta_3}$ donné partie I, page 45.

4.2.4 Diagramme associé à la spécification A_4

Rappelons la définition de la spécification A_4 des anneaux. Cette spécification utilise la spécification B_2 que nous avons déjà vue dans le paragraphe précédent, ainsi qu'une autre spécification P' des pseudo-anneaux.

$$\begin{aligned}B_2 &\equiv \mathsf{push}\,(\emptyset, B, B, \mathrm{j}_B, \mathrm{j}_B)\\ u_2 &\equiv \mathsf{up}\,(\emptyset, B, B, D, \mathrm{j}_B, \mathrm{j}_B, m_*, m_+) : B_2 \to D\end{aligned}$$

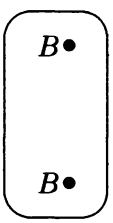
$$\begin{aligned}Q_1 &\equiv \mathsf{push}\,(S, B, B, s, s)\\ Q_2 &\equiv \mathsf{push}\,(B, M, Q_1, b, \&_1(Q_1))\\ P' &\equiv \mathsf{push}\,(B, Q_2, G, \&_2(Q_2) \circ \&_2(Q_1), m \circ b)\end{aligned}$$

$$\begin{aligned}u_3 \equiv \mathsf{up}\,(&\emptyset, B, B, P', \mathrm{j}_B, \mathrm{j}_B, \&_1(P') \circ \&_2(Q_2) \circ \&_1(Q_1),\\ &\&_1(P') \circ \&_2(Q_2) \circ \&_2(Q_1)) : B_2 \to P'\end{aligned}$$

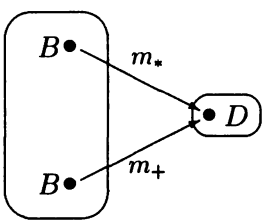
$$A_4 \equiv \mathsf{push}\,(B_2, P', D, u_3, u_2)$$

Nous avons vu dans le paragraphe précédent le diagramme associé à la spécification B_2, ainsi que le morphisme de diagrammes associé au morphisme de spécifications $u_2 : B_2 \to D$.

$\mathcal{D}(B_2) \equiv$

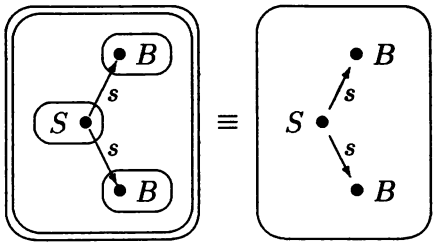


$\mathcal{D}(u_2) \equiv$

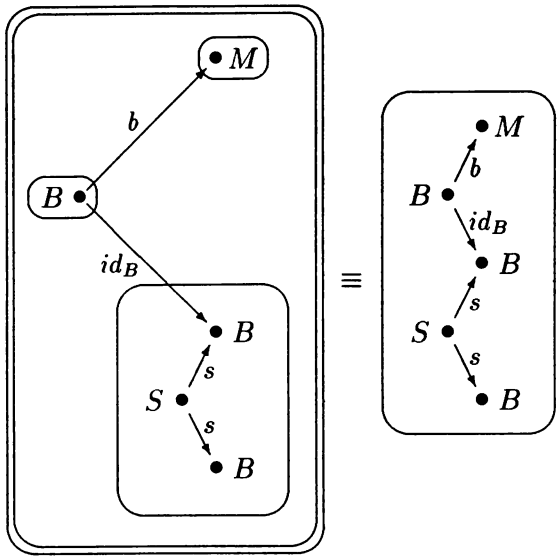


Calculons maintenant les diagrammes associés aux spécifications Q_1, Q_2 et P'.

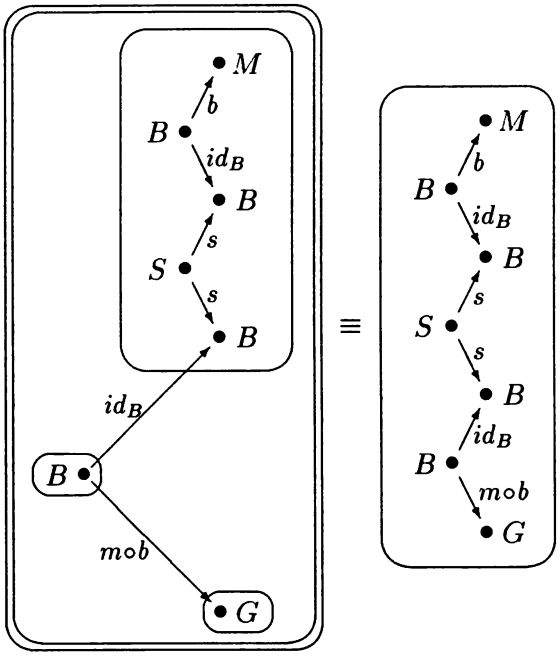
$\mathcal{D}(Q_1) \equiv \mathrm{Apl}_{\mathcal{C}_0}$



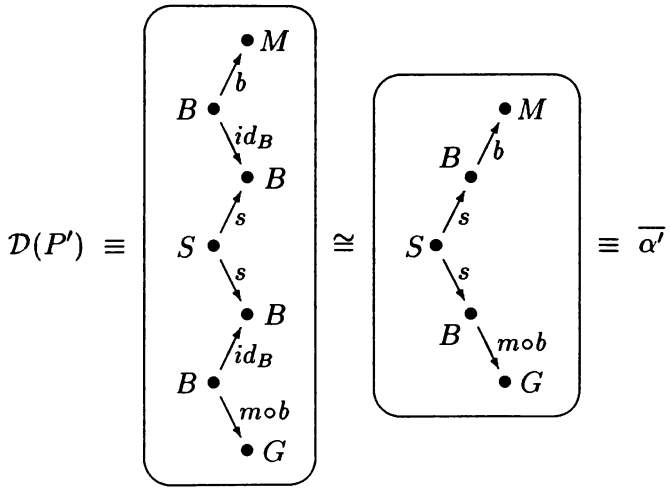
$\mathcal{D}(Q_2) \equiv \mathrm{Apl}_{\mathcal{C}_0}$



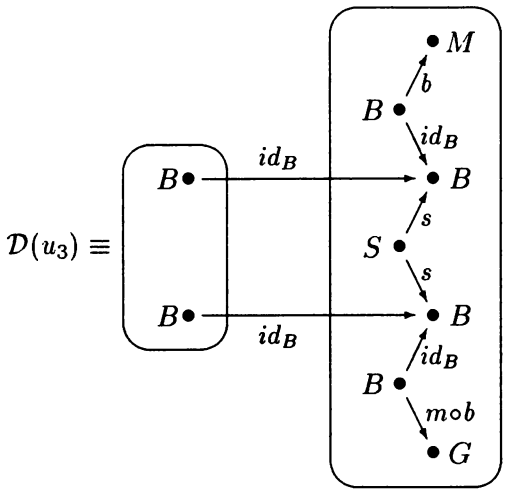
$\mathcal{D}(P') \equiv \mathrm{Apl}_{\mathcal{C}_0}$



Si un diagramme $\overline{\delta}$ contient un arc $a : n \to n'$ étiqueté par la flèche identité (c'est-à-dire tel que $\delta(n) = \delta(n')$ et $\delta(a) = id_{\alpha(n)}$), alors on obtient un diagramme isomorphe en en fusionnant les nœuds n et n' et en supprimant l'arc $a : n \to n'$. Ce résultat sera démontré dans le chapitre 5. Dans notre exemple, le diagramme $\mathcal{D}(P')$ est donc isomorphe au diagramme $\overline{\alpha'}$ ci-dessous.



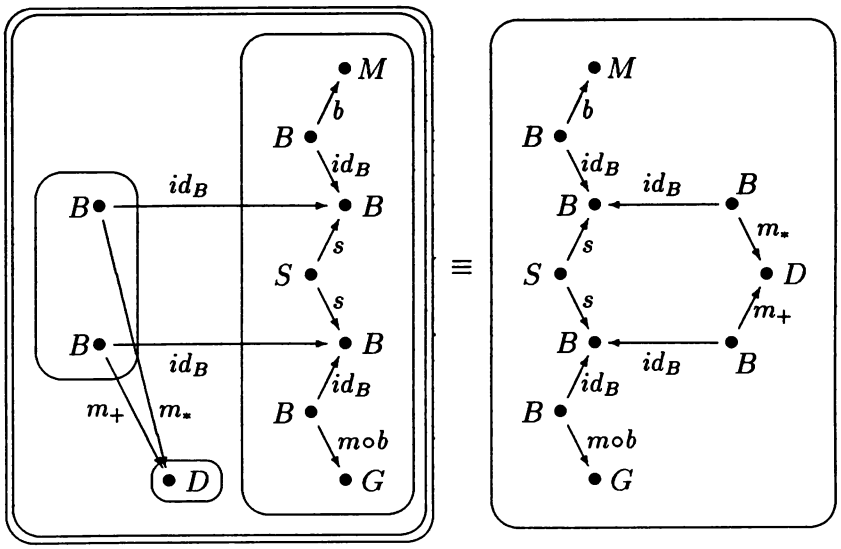
Revenons au calcul de $\mathcal{D}(A_4)$. Il reste à calculer le morphisme de diagrammes associé au morphisme de spécifications $u_3 : B_2 \to P'$.



On en déduit enfin le diagramme associé à la spécification A_4.

$\mathcal{D}(A_4) \equiv \mathcal{D}(\textsf{push}\,(B_2, P', D, u_3, u_2))$

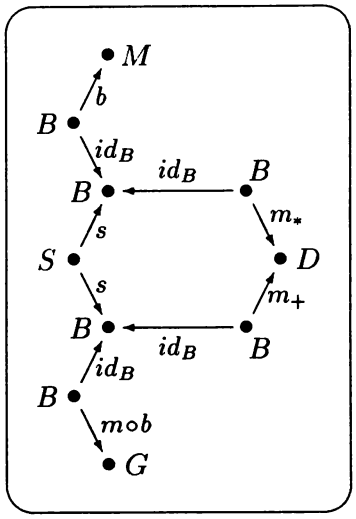
$\equiv \mathrm{Apl}_{\mathcal{C}_0}$



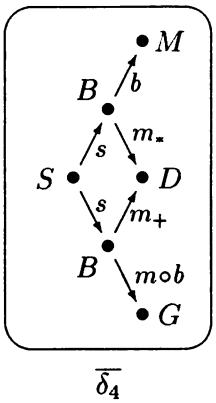
$\equiv$

En supprimant les quatre arcs étiquetés par des flèches identités, on obtient un diagramme $\overline{\delta_4}$, isomorphe à $\mathcal{D}(A_4)$.

$\mathcal{D}(A_4) \equiv$



$\cong$



$\overline{\delta_4}$

$\equiv \overline{\delta_4}$

4.3 Équivalence entre les catégories Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$)

Dans cette partie, nous montrons que les catégories Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$) sont équivalentes. Pour cela nous montrons en fait que les *précatégories* TERME($\mathcal{C}_0$) et DIAGR($\mathcal{C}_0$) sont équivalentes.

4.3.1 Équivalence entre catégories, équivalence entre précatégories

Nous commençons par définir la notion de *transformation naturelle entre préfoncteurs.*

Définition 4.1 (Transformation naturelle entre deux préfoncteurs) Soit deux précatégories $\mathcal{C}$ et $\mathcal{C}'$. Soit deux préfoncteurs $F, G : \mathcal{C} \to \mathcal{C}'$. Une *transformation naturelle* σ de F vers G, notée $\sigma : F \dot{\to} G$, est une application qui à tout objet A de $\mathcal{C}$ associe une flèche $\sigma_A \in \mathrm{Arr}(\mathcal{C}')(F(A), G(A))$, telle que pour toute flèche $f \in \mathrm{Arr}(\mathcal{C})(A, B)$,

$$G(f) \circ \sigma_A = \sigma_B \circ F(f) \in \mathrm{Arr}(\mathcal{C}')(F(A), G(B)).$$

Par conséquent, σ est une transformation naturelle entre les deux préfoncteurs F et G si et seulement si σ est une transformation naturelle entre les foncteurs F et G.

Définition 4.2 (Isomorphisme naturel entre deux préfoncteurs) Une transformation naturelle $\sigma : F \dot{\to} G$ entre deux préfoncteurs $F, G : \mathcal{C} \to \mathcal{C}'$ est un *isomorphisme naturel* si et seulement si pour tout objet A de $\mathcal{C}$, $\sigma_A \in \mathrm{Arr}(\mathcal{C}')(F(A), G(A))$ est un isomorphisme.

Si $\sigma : F \dot{\to} G$ est un isomorphisme naturel, on dit que les préfoncteurs F et G sont *naturellement isomorphes*, et on note $F \cong G$. On vérifie facilement que deux préfoncteurs sont naturellement isomorphes si et seulement si les foncteurs associés sont naturellement isomorphes.

Rappelons la définition d'*équivalence* entre deux catégories.

Définition 4.3 (Équivalence) Deux catégories $\mathcal{C}$ et $\mathcal{C}'$ sont *équivalentes* si et seulement si il existe deux foncteurs $F : \mathcal{C} \to \mathcal{C}'$ et $G : \mathcal{C}' \to \mathcal{C}$ tels qu'on ait deux isomorphismes naturels $G \circ F \cong Id_{\mathcal{C}}$ et $F \circ G \cong Id_{\mathcal{C}'}$.

On en déduit une définition d'*équivalence* entre deux *précatégories.*

Définition 4.4 Deux précatégories $\mathcal{C}$ et $\mathcal{C}'$ sont *équivalentes* si et seulement si il existe deux préfoncteurs $F : \mathcal{C} \to \mathcal{C}'$ et $G : \mathcal{C}' \to \mathcal{C}$ tels qu'on ait deux isomorphismes naturels $G \circ F \cong Id_{\mathcal{C}}$ et $F \circ G \cong Id_{\mathcal{C}'}$.

Deux précatégories sont équivalentes si et seulement si les catégories associées sont équivalentes.

4.3.2 Équivalence entre les précatégories DIAGR($\mathcal{C}_0$) et TERME($\mathcal{C}_0$)

Dans ce paragraphe, nous montrons qu'il existe une équivalence entre les précatégories DIAGR($\mathcal{C}_0$) et TERME($\mathcal{C}_0$). Intuitivement, cela signifie que ces deux précatégories ont "la même structure", "aux isomorphismes entre objets près".

Nous commençons par reformuler le théorème 2.7 (partie I, page 109) en utilisant des précatégories.

Théorème 4.1 *La précatégorie* DIAGR($\mathcal{C}_0$) *est une complétion de* $\mathcal{C}_0$ *par pré-colimites finies. Autrement dit, pour toute précatégorie* $\mathcal{E}$ *finiment pré-cocomplète, et tout préfoncteur* $F : \mathcal{C}_0 \to \mathcal{E}$, *il existe un préfoncteur*

$$H : \mathrm{DIAGR}(\mathcal{C}_0) \to \mathcal{E},$$

unique à un isomorphisme naturel près, qui conserve les pré-colimites finies, tel que

$$H \circ I_{\mathcal{C}_0} \cong F.$$

Comme la précatégorie TERME($\mathcal{C}_0$) est finiment pré-cocomplète, et comme on a un préfoncteur $J : \mathcal{C}_0 \to \text{TERME}(\mathcal{C}_0)$, il existe un préfoncteur

$$\mathcal{T} : \text{DIAGR}(\mathcal{C}_0) \to \text{TERME}(\mathcal{C}_0),$$

qui conserve les pré-colimites finies, tel que

$$\mathcal{T} \circ I_{\mathcal{C}_0} \cong J.$$

D'autre part, nous avons vu au paragraphe 4.1.2 que le préfoncteur

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

conserve l'objet pré-initial et les pré-sommes amalgamées. Le préfoncteur $\mathcal{D}$ conserve donc les pré-colimites finies. De plus, on a

$$\mathcal{D} \circ J \equiv I_{\mathcal{C}_0}.$$

Lemme 4.3 *On a les isomorphismes naturels suivants:*

1. $\mathcal{D} \circ \mathcal{T} \circ I_{\mathcal{C}_0} \cong I_{\mathcal{C}_0}$;
2. $\mathcal{T} \circ \mathcal{D} \circ J \cong J$.

Preuve. C'est une conséquence immédiate de $\mathcal{T} \circ I_{\mathcal{C}_0} \cong J$ et $\mathcal{D} \circ J \equiv I_{\mathcal{C}_0}$. □

Proposition 4.1 *On a un isomorphisme naturel*

$$\mathcal{D} \circ \mathcal{T} \cong Id_{\text{DIAGR}(\mathcal{C}_0)}.$$

Preuve. D'après le lemme 4.3–1, on a $\mathcal{D} \circ \mathcal{T} \circ I_{\mathcal{C}_0} \cong I_{\mathcal{C}_0}$. De plus, les préfoncteurs $\mathcal{D} \circ \mathcal{T} : \text{DIAGR}(\mathcal{C}_0) \to \text{DIAGR}(\mathcal{C}_0)$ et $Id_{\text{DIAGR}(\mathcal{C}_0)} : \text{DIAGR}(\mathcal{C}_0) \to \text{DIAGR}(\mathcal{C}_0)$ conservent les pré-colimites finies. Par conséquent, d'après le théorème 4.1,

$$\mathcal{D} \circ \mathcal{T} \cong Id_{\text{DIAGR}(\mathcal{C}_0)}.$$

□

Lemme 4.4 *Soit* $A, B, C \in \text{Obj}(\text{TERME}(\mathcal{C}_0))$, $f \in \text{Arr}(\text{TERME}(\mathcal{C}_0))(A, B)$ *et* $g \in \text{Arr}(\text{TERME}(\mathcal{C}_0))(A, C)$. *Alors, le triplet*

$$\begin{array}{ll} (& (\mathcal{T} \circ \mathcal{D})(\textsf{push}\,(A, B, C, f, g)), \\ & (\mathcal{T} \circ \mathcal{D})(\&_1(A, B, C, f, g)), \\ & (\mathcal{T} \circ \mathcal{D})(\&_2(A, B, C, f, g)) \quad) \end{array}$$

est une pré-somme amalgamée de $(\mathcal{T} \circ \mathcal{D})(B)$ *et* $(\mathcal{T} \circ \mathcal{D})(C)$ *par rapport aux flèches* $(\mathcal{T} \circ \mathcal{D})(f)$ *et* $(\mathcal{T} \circ \mathcal{D})(g)$.

Preuve. Comme les préfoncteurs $\mathcal{D}$ et $\mathcal{T}$ conservent les pré-sommes amalgamées, le préfoncteur $\mathcal{T} \circ \mathcal{D}$ conserve également les pré-sommes amalgamées. □

Proposition 4.2 *On a un isomorphisme naturel*

$$\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\mathrm{TERME}(\mathcal{C}_0)}.$$

Preuve. Comme $\mathcal{T} \circ \mathcal{D}$ et $Id_{\mathrm{TERME}(\mathcal{C}_0)}$ sont des préfoncteurs de la précatégorie TERME($\mathcal{C}_0$) vers TERME($\mathcal{C}_0$), il faut donner, pour tout objet U de TERME($\mathcal{C}_0$), un isomorphisme

$$\Psi_U : (\mathcal{T} \circ \mathcal{D})(U) \to U$$

dans TERME($\mathcal{C}_0$). On définit cet isomorphisme par induction sur la structure des objets de TERME($\mathcal{C}_0$), et on vérifie qu'il s'agit bien d'une transformation naturelle par induction sur la structure des flèches de TERME($\mathcal{C}_0$).

Soit un objet U de TERME($\mathcal{C}_0$). On définit Ψ_U par induction sur la structure des objets de TERME($\mathcal{C}_0$) (cf. partie II, règles 1 à 10, page 22).

Règle (1) $U \equiv J(A)$, où A est un objet de $\mathcal{C}_0$.

D'après le lemme 4.3–2, $\mathcal{T} \circ \mathcal{D} \circ J \cong J$. Par conséquent, il existe un isomorphisme

$$\Psi_U : (\mathcal{T} \circ \mathcal{D})(U) \to U.$$

Règle (2) $U \equiv \emptyset$.

$(\mathcal{T} \circ \mathcal{D})(\emptyset) \equiv \mathcal{T}(\bigcirc) \cong \emptyset$ car $\mathcal{T}$ conserve l'objet pré-initial. Soit

$$\Psi_\emptyset : (\mathcal{T} \circ \mathcal{D})(\emptyset) \to \emptyset$$

cet isomorphisme.

Règle (3) $U \equiv \mathsf{push}\,(A, B, C, f, g)$.

Par hypothèse d'induction, on a trois isomorphismes

$$\Psi_A : (\mathcal{T} \circ \mathcal{D})(A) \to A$$
$$\Psi_B : (\mathcal{T} \circ \mathcal{D})(B) \to B$$
$$\Psi_C : (\mathcal{T} \circ \mathcal{D})(C) \to C$$

tels que

$$f \circ \Psi_A = \Psi_B \circ (\mathcal{T} \circ \mathcal{D})(f)$$
$$g \circ \Psi_A = \Psi_C \circ (\mathcal{T} \circ \mathcal{D})(g).$$

D'après le lemme 4.4, le triplet

$$\begin{pmatrix} (\mathcal{T} \circ \mathcal{D})(\mathsf{push}\,(A, B, C, f, g)), \\ (\mathcal{T} \circ \mathcal{D})(\&_1(A, B, C, f, g)), \\ (\mathcal{T} \circ \mathcal{D})(\&_2(A, B, C, f, g)) \end{pmatrix}$$

est une pré-somme amalgamée de $(\mathcal{T} \circ \mathcal{D})(B)$ et $(\mathcal{T} \circ \mathcal{D})(C)$ par rapport aux flèches $(\mathcal{T} \circ \mathcal{D})(f)$ et $(\mathcal{T} \circ \mathcal{D})(g)$.

D'autre part, le triplet

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

est évidemment une pré-somme amalgamée de B et C par rapport aux flèches f et g dans TERME($\mathcal{C}_0$).

Par conséquent, il existe un unique isomorphisme

$$\Psi_U = \Psi_{\mathsf{push}(A,B,C,f,g)} : (\mathcal{T} \circ \mathcal{D})(\mathsf{push}\,(A,B,C,f,g)) \to \mathsf{push}\,(A,B,C,f,g)$$

tel que

$$\Psi_U \circ (\mathcal{T} \circ \mathcal{D})(\&_1(A,B,C,f,g)) = \&_1(A,B,C,f,g) \circ \Psi_B$$
$$\Psi_U \circ (\mathcal{T} \circ \mathcal{D})(\&_2(A,B,C,f,g)) = \&_2(A,B,C,f,g) \circ \Psi_C.$$

On vérifie maintenant qu'on définit bien une *transformation naturelle*

$$\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\mathrm{TERME}(\mathcal{C}_0)},$$

par induction sur la structure des flèches de TERME($\mathcal{C}_0$).

Règle (4) Soit $f : A \to B$ une flèche de $\mathcal{C}_0$. Comme Ψ_A et Ψ_B sont les isomorphismes correspondant à l'isomorphisme naturel $\mathcal{T} \circ \mathcal{D} \circ J \cong J$, on a donc

$$\Psi_B \circ (\mathcal{T} \circ \mathcal{D})(f) = f \circ \Psi_A.$$

Règle (5) Soit deux flèches $f : A \to B$ et $g : B \to C$ de TERME($\mathcal{C}_0$).

$$\begin{array}{lll} \Psi_C \circ (\mathcal{T} \circ \mathcal{D})(g \circ f) & & \\ \quad = \Psi_C \circ (\mathcal{T} \circ \mathcal{D})(g) \circ (\mathcal{T} \circ \mathcal{D})(f) & & (\mathcal{T} \circ \mathcal{D} \text{ préfoncteur}) \\ \quad = g \circ \Psi_B \circ (\mathcal{T} \circ \mathcal{D})(f) & & (\text{hypothèse d'induction sur } g) \\ \quad = g \circ f \circ \Psi_A & & (\text{hypothèse d'induction sur } f) \end{array}$$

Règle (6) $\Psi_A \circ (\mathcal{T} \circ \mathcal{D})(\mathsf{id}_A) = \mathsf{id}_A \circ \Psi_A$

Règle (7) $\Psi_A \circ (\mathcal{T} \circ \mathcal{D})(\mathsf{j}_A) = \mathsf{j}_A \circ \Psi_\emptyset$ car ces deux flèches ont pour domaine un objet pré-initial de TERME($\mathcal{C}_0$).

Règle (8) Par définition de $\Psi_{\mathsf{push}(A,B,C,f,g)}$, on a

$$\Psi_{\mathsf{push}(A,B,C,f,g)} \circ (\mathcal{T} \circ \mathcal{D})(\&_1(A,B,C,f,g)) = \&_1(A,B,C,f,g) \circ \Psi_B.$$

Règle (9) Par définition de $\Psi_{\mathsf{push}(A,B,C,f,g)}$, on a

$$\Psi_{\mathsf{push}(A,B,C,f,g)} \circ (\mathcal{T} \circ \mathcal{D})(\&_2(A,B,C,f,g)) = \&_2(A,B,C,f,g) \circ \Psi_C.$$

Règle (10) Posons

$$\begin{aligned}\&_1 &= \&_1(A,B,C,f,g)\\ \&_2 &= \&_2(A,B,C,f,g)\\ \mathsf{up} &= \mathsf{up}\,(A,B,C,D,f,g,f',g')\\ \Psi_{\mathsf{push}} &= \Psi_{\mathsf{push}(A,B,C,f,g)}\end{aligned}$$

Il faut montrer que

$$\Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}) = \mathsf{up}\circ\Psi_{\mathsf{push}}.$$

Pour cela, il suffit de montrer

$$\begin{aligned}\Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}) \circ (\mathcal{T}\circ\mathcal{D})(\&_1) &= \mathsf{up}\circ\Psi_{\mathsf{push}}\circ(\mathcal{T}\circ\mathcal{D})(\&_1)\\ \Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}) \circ (\mathcal{T}\circ\mathcal{D})(\&_2) &= \mathsf{up}\circ\Psi_{\mathsf{push}}\circ(\mathcal{T}\circ\mathcal{D})(\&_2).\end{aligned}$$

$$\begin{aligned}&\Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}) \circ (\mathcal{T}\circ\mathcal{D})(\&_1)\\ &= \Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}\circ\&_1) && (\mathcal{T}\circ\mathcal{D}\text{ préfoncteur})\\ &= \Psi_D \circ (\mathcal{T}\circ\mathcal{D})(f') && (\text{définition de up})\\ &= f'\circ\Psi_B && (\text{hypothèse d'induction sur } f')\\ &= \mathsf{up}\circ\&_1\circ\Psi_B && (\text{définition de up})\\ &= \mathsf{up}\circ\Psi_{\mathsf{push}}\circ(\mathcal{T}\circ\mathcal{D})(\&_1) && (\text{règle 8})\end{aligned}$$

On montre de la même façon que

$$\Psi_D \circ (\mathcal{T}\circ\mathcal{D})(\mathsf{up}) \circ (\mathcal{T}\circ\mathcal{D})(\&_2) = \mathsf{up}\circ\Psi_{\mathsf{push}}\circ(\mathcal{T}\circ\mathcal{D})(\&_2).$$

□

Théorème 4.2 *Les précatégories* TERME($\mathcal{C}_0$) *et* DIAGR($\mathcal{C}_0$) *sont équivalentes.*

Preuve. Conséquence immédiate des propositions 4.1 et 4.2. □

Corollaire 4.1 *Les catégories* Terme($\mathcal{C}_0$) *et* diagr($\mathcal{C}_0$) *sont équivalentes.*

La proposition 4.2 nous permet de montrer que deux objets de Terme($\mathcal{C}_0$) sont isomorphes si et seulement si leurs diagrammes associés sont isomorphes, et deux flèches de Terme($\mathcal{C}_0$) sont égales si et seulement si leurs morphismes de diagrammes associés sont égaux dans diagr($\mathcal{C}_0$).

Théorème 4.3 *Considérons le foncteur* $\mathcal{D}$: Terme($\mathcal{C}_0$) $\to$ diagr($\mathcal{C}_0$).

- *Soit A et B deux objets de* Terme($\mathcal{C}_0$). *Alors,*
$$A \cong B \;\Leftrightarrow\; \mathcal{D}(A) \cong \mathcal{D}(B).$$
- *Soit $f, g : A \to B$ deux flèches de* Terme($\mathcal{C}_0$). *Alors,*
$$f = g \;\Leftrightarrow\; \mathcal{D}(f) = \mathcal{D}(g).$$

Preuve. Les deux implications de gauche à droite sont évidentes car $\mathcal{D}$ est un foncteur. Dans l'autre sens, d'après la proposition 4.2, il existe un isomorphisme naturel $\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\text{Terme}(\mathcal{C}_0)}$ entre les foncteurs $\mathcal{T} \circ \mathcal{D} : \text{Terme}(\mathcal{C}_0) \to \text{Terme}(\mathcal{C}_0)$ et $Id_{\text{Terme}(\mathcal{C}_0)} : \text{Terme}(\mathcal{C}_0) \to \text{Terme}(\mathcal{C}_0)$.

- Soit deux objets A et B de $\text{Terme}(\mathcal{C}_0)$.

$$\begin{array}{lll} \mathcal{D}(A) \cong \mathcal{D}(B) & & \\ \quad \Rightarrow \quad (\mathcal{T} \circ \mathcal{D})(A) \cong (\mathcal{T} \circ \mathcal{D})(B) & & \\ \quad \Rightarrow \quad A \cong B & & (\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\text{Terme}(\mathcal{C}_0)}) \end{array}$$

- Soit deux flèches $f, g : A \to B$ de $\text{Terme}(\mathcal{C}_0)$ telles que $\mathcal{D}(f) = \mathcal{D}(g)$ dans $\text{diagr}(\mathcal{C}_0)$.

$$\begin{array}{lll} \mathcal{D}(f) = \mathcal{D}(g) & & \\ \quad \Rightarrow \quad (\mathcal{T} \circ \mathcal{D})(f) = (\mathcal{T} \circ \mathcal{D})(g) & & \\ \quad \Rightarrow \quad \Psi_B \circ (\mathcal{T} \circ \mathcal{D})(f) = \Psi_B \circ (\mathcal{T} \circ \mathcal{D})(g) & & \\ \quad \Rightarrow \quad f \circ \Psi_A = g \circ \Psi_A & & (\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\text{Terme}(\mathcal{C}_0)}) \\ \quad \Rightarrow \quad f = g & & (\Psi_A \text{ isomorphisme}) \end{array}$$

□

4.3.3 Correction du calcul de diagrammes

La proposition 4.2 nous permet également de déduire que le calcul de diagrammes décrit par le foncteur $\mathcal{D} : \text{Terme}(\mathcal{C}_0) \to \text{diagr}(\mathcal{C}_0)$ est *correct* c'est-à-dire que la colimite d'un diagramme associé à une spécification modulaire est isomorphe à cette spécification.

Considérons une catégorie d'interprétation $\mathcal{E}$ à objet initial choisi et sommes amalgamées choisies, par exemple la catégorie des spécifications **Spec**. Soit une interprétation de $\mathcal{C}_0$, c'est-à-dire un foncteur

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

D'après le théorème 2.7, ce foncteur s'étend aux diagrammes en un foncteur

$$\mathcal{S}_D : \text{diagr}(\mathcal{C}_0) \to \mathcal{E}.$$

On a en fait $\mathcal{S}_D = \text{colim}_{\mathcal{E}} \circ \text{diagr}(\mathcal{S}_0)$.
D'après le théorème 3.9, le foncteur $\mathcal{S}_0$ s'étend également aux termes en un foncteur

$$\mathcal{S}_T : \text{Terme}(\mathcal{C}_0) \to \mathcal{E}.$$

De plus, comme $\mathcal{S}_D$ et $\mathcal{S}_T$ conservent les colimites, d'après le théorème 2.7, on a

$$\mathcal{S}_D \cong \mathcal{S}_T \circ \mathcal{T}.$$

Par conséquent, $\mathcal{S}_D \circ \mathcal{D} \cong \mathcal{S}_T \circ \mathcal{T} \circ \mathcal{D}$, et donc, d'après la proposition 4.2,

$$\mathcal{S}_D \circ \mathcal{D} \cong \mathcal{S}_T.$$

Ce résultat exprime que la colimite d'un diagramme associé à une spécification modulaire est isomorphe à cette spécification.

4.4 Conclusion

Dans ce chapitre, nous avons montré comment associer à tout terme qui représente une spécification modulaire un diagramme dont la colimite est isomorphe à cette spécification. Cette association est décrite par le préfoncteur

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

Le calcul d'un diagramme associé à un terme consiste simplement à appliquer les règles de calcul de $\mathcal{D}$ définies dans le paragraphe 4.1.2. Une des règles principales est la règle 3 qui *aplatit* un diagramme de diagrammes, c'est-à-dire transforme un objet de $\mathrm{DIAGR}^2(\mathcal{C}_0)$ en un objet de $\mathrm{DIAGR}(\mathcal{C}_0)$.

Nous avons montré que les précatégories $\mathrm{TERME}(\mathcal{C}_0)$ et $\mathrm{DIAGR}(\mathcal{C}_0)$ sont équivalentes, ce qui signifie qu'elles ont essentiellement la même structure "aux isomorphismes entre objets près". Il faut néanmoins remarquer que $\mathrm{TERME}(\mathcal{C}_0)$ et $\mathrm{DIAGR}(\mathcal{C}_0)$ *ne sont pas isomorphes.* En fait, pour qu'elles soient isomorphes, il faudrait que ces deux précatégories aient les mêmes choix de pré-colimites, ce qui n'est pas le cas. Par exemple, $\mathbf{push}\,(A, B, C, f, g)$ et $\mathbf{push}\,(A, C, B, g, f)$ sont deux présommes amalgamées différentes dans $\mathrm{TERME}(\mathcal{C}_0)$ qui sont envoyées sur le même diagramme dans $\mathrm{DIAGR}(\mathcal{C}_0)$.

Chapitre 5

Isomorphismes entre diagrammes

Nous considérons une catégorie de base $\mathcal{C}_0$, et nous nous intéressons dans ce chapitre à la détection d'isomorphismes entre deux diagrammes dans la catégorie diagr($\mathcal{C}_0$). Nous avons vu dans le chapitre 4 que deux objets A et B de la catégorie Terme($\mathcal{C}_0$) sont isomorphes si et seulement si leurs diagrammes associés $\mathcal{D}(A)$ et $\mathcal{D}(B)$ sont isomorphes (théorème 4.3 page 31). Une solution pour savoir si deux termes A et B sont isomorphes consiste donc à détecter un isomorphisme entre leurs diagrammes associés.

En section 5.1, nous décrivons des transformations de diagrammes, qui sont des fonctions sur l'ensemble des diagrammes sur $\mathcal{C}_0$. Les transformations qui nous intéressent particulièrement sont les transformations *stables par isomorphisme*, c'est-à-dire telles que le diagramme résultat est isomorphe au diagramme de départ. Ces transformations nous serviront à calculer la *complétion* d'un diagramme.

En section 5.2, nous présentons un algorithme pour détecter si deux diagrammes sont isomorphes, dans la cas particulier où la catégorie $\mathcal{C}_0$ et finie et sans cycle. Cet algorithme consiste à compléter puis à calculer la *forme minimale* du diagramme. Ensuite, deux diagrammes sont isomorphes si et seulement si ils ont la même forme minimale.

Enfin, en section 5.3, nous montrons les difficultés rencontrées lorsque la catégorie de base $\mathcal{C}_0$ comporte des cycles ou est infinie.

5.1 Transformations de diagrammes

Nous présentons quelques transformations de diagrammes: la *substitution par objet isomorphe*, la *suppression d'un arc*, l'*ajout d'un arc étiqueté* et la *contraction d'un arc identité*.

Définition 5.1 (Transformation de diagrammes) Une transformation de diagrammes est une fonction sur l'ensemble des objets de diagr($\mathcal{C}_0$)

$$\mathcal{F} : \text{diagr}(\mathcal{C}_0) \rightarrow \text{diagr}(\mathcal{C}_0).$$

Les transformations intéressantes sont les transformations *stables par isomorphisme* c'est-à-dire qui produisent un diagramme isomorphe au diagramme de départ.

Définition 5.2 (Transformation stable par isomorphisme) Une transformation de diagrammes $\mathcal{F} : \text{diagr}(\mathcal{C}_0) \to \text{diagr}(\mathcal{C}_0)$ est *stable par isomorphisme* si et seulement si pour tout diagramme $\overline{\delta}$,

$$\mathcal{F}(\overline{\delta}) \cong \overline{\delta}.$$

Deux diagrammes $\overline{\alpha}$ et $\overline{\beta}$ sur $\mathcal{C}_0$ sont isomorphes dans $\text{diagr}(\mathcal{C}_0)$ s'il existe deux flèches $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\alpha}$ de $\text{DIAGR}(\mathcal{C}_0)$ telles que

$$\begin{cases} [\overline{\sigma}] \circ [\overline{\tau}] = [\overline{id_\beta}] \\ [\overline{\tau}] \circ [\overline{\sigma}] = [\overline{id_\alpha}] \end{cases} \Leftrightarrow \begin{cases} \overline{\sigma} \circ \overline{\tau} \approx \overline{id_\beta} \\ \overline{\tau} \circ \overline{\sigma} \approx \overline{id_\alpha}. \end{cases}$$

Rappelons qu'une flèche $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ de $\text{DIAGR}(\mathcal{C}_0)$ est un couple (cf. définition 2.18, partie I, page 66)

$$\overline{\sigma} : \overline{\alpha} \to \overline{\beta} = (\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi,\ \sigma : \alpha \rightsquigarrow \beta \circ \sigma^\Phi)$$

où $\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$ est un morphisme généralisé de graphes et $\sigma : \alpha \rightsquigarrow \beta \circ \sigma^\Phi$ est une transformation naturelle généralisée.

- Le morphisme généralisé de graphes

  $$\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$$

  associe à tout nœud de α^Φ un nœud de β^Φ, et à tout arc $a : m \to n$ de α^Φ un *zigzag* $\sigma^\Phi(a) : \sigma^\Phi(m) \rightsquigarrow \sigma^\Phi(n)$ sur le graphe β^Φ.

- La transformation naturelle généralisée $\sigma : \alpha \rightsquigarrow \beta \circ \sigma^\Phi$ associe à tout nœud n de α^Φ une flèche $\sigma_n : \alpha(n) \to \beta(\sigma^\Phi(n))$. De plus, la condition suivante doit être satisfaite : pour tout arc $a : m \to n$ de α^Φ,

  $$\sigma_n \circ \alpha(a) \sim_{\overline{\beta}} \sigma_m \ \ [\sigma^\Phi(a)].$$

5.1.1 Substitution par objet isomorphe

Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$, un nœud n_0 de δ^Φ, et un objet B de $\mathcal{C}_0$ isomorphe à $\delta(n_0)$. Intuitivement, la *substitution par objet isomorphe* consiste à remplacer l'étiquette $\delta(n_0)$ du nœud n_0 par l'objet B. On obtient alors un diagramme isomorphe au diagramme de départ $\overline{\delta}$.

Définition 5.3 (Subst_Obj_Iso) Soit un diagramme $\overline{\delta}$, un nœud n_0 de δ^Φ, et un objet B de $\mathcal{C}_0$ isomorphe à $\delta(n_0)$. Soit $\phi : \delta(n_0) \to B$ l'isomorphisme entre $\delta(n_0)$ et B. Le diagramme

$$\beta = \text{Subst\_Obj\_Iso}(\overline{\delta}, n_0, B, \phi)$$

est défini de la façon suivante.

- Graphe β^Φ : $\beta^\Phi = \delta^\Phi$.

- Foncteur $\beta : \beta^\Phi \to \mathcal{C}_0$.
  - Action sur les nœuds.
    - $\forall n \in \text{Nœud}(\delta^\Phi),\ n \neq n_0 : \beta(n) = \delta(n)$;
    - $\beta(n_0) = B$.
  - Action sur les arcs. Soit un arc $a : n \to n' \in \text{Arc}(\delta^\Phi)$.
    - si $n \neq n_0$ et $n' \neq n_0 : \beta(a) = \delta(a)$;
    - si $n = n_0$ et $n' \neq n_0 : \beta(a) = \delta(a) \circ \phi^{-1}$;
    - si $n \neq n_0$ et $n' = n_0 : \beta(a) = \phi \circ \delta(a)$;
    - si $n = n_0$ et $n' = n_0 : \beta(a) = \phi \circ \delta(a) \circ \phi^{-1}$.

Proposition 5.1 *La transformation* Subst_Obj_Iso *est stable par isomorphisme.*

Preuve. Soit un diagramme $\overline{\delta}$, un nœud n_0 de δ^Φ, et un isomorphisme $\phi : \delta(n_0) \to B$ de $\mathcal{C}_0$. Soit $\overline{\beta} = \text{Subst\_Obj\_Iso}(\overline{\delta}, n_0, B, \phi)$. On construit deux flèches de $\text{DIAGR}(\mathcal{C}_0)$, $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id}_{\overline{\beta}}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id}_{\overline{\delta}}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Le morphisme généralisé de graphes $\sigma^\Phi : \delta^\Phi \to \beta^\Phi$ est l'identité.
    - $\sigma^\Phi = id_{\delta^\Phi}$.
  - La transformation naturelle généralisée $\sigma : \delta \dashrightarrow \beta \circ \sigma^\Phi$ est définie par :
    - $\forall n \in \text{Nœud}(\delta^\Phi)$ tel que $n \neq n_0$, $\sigma_n = id_{\delta(n)}$;
    - $\sigma_{n_0} = \phi$.

    Cela définit bien une transformation naturelle généralisée car pour tout arc $a : n \to n' \in \text{Arc}(\delta^\Phi)$, on a $\sigma_{n'} \circ \delta(a) = \beta(a) \circ \sigma_n$.

- On définit $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ de la façon suivante.
  - Le morphisme généralisé de graphes $\tau^\Phi : \beta^\Phi \to \delta^\Phi$ est l'identité.
    - $\tau^\Phi = id_{\beta^\Phi}$.
  - La transformation naturelle généralisée $\tau : \beta \dashrightarrow \delta \circ \tau^\Phi$ est définie par :
    - $\forall n \in \text{Nœud}(\beta^\Phi)$ tel que $n \neq n_0$, $\tau_n = id_{\beta(n)}$;
    - $\tau_{n_0} = \phi^{-1}$.

    Cela définit bien une transformation naturelle généralisée car pour tout arc $a : n \to n' \in \text{Arc}(\beta^\Phi)$, on a $\tau_{n'} \circ \beta(a) = \delta(a) \circ \tau_n$.

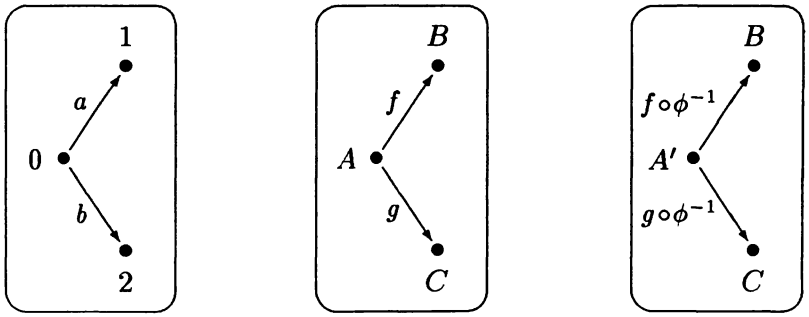
- On vérifie que $\overline{\sigma} \circ \overline{\tau} = \overline{id}_{\overline{\beta}}$.
- On vérifie également que $\overline{\tau} \circ \overline{\sigma} = \overline{id}_{\overline{\delta}}$.

Par conséquent $\overline{\delta} \cong \overline{\beta}$ dans la catégorie $\text{diagr}(\mathcal{C}_0)$. En fait, les diagrammes $\overline{\delta}$ et $\overline{\beta}$ sont ici également isomorphes dans la catégorie $\text{DIAGR}(\mathcal{C}_0)$.

□

Exemple 5.1 Considérons le diagramme $\overline{\pi}$, construit sur le graphe π^Φ. Soit $\phi : A \to A'$ un isomorphisme de $\mathcal{C}_0$. Alors, nous pouvons considérer le diagramme

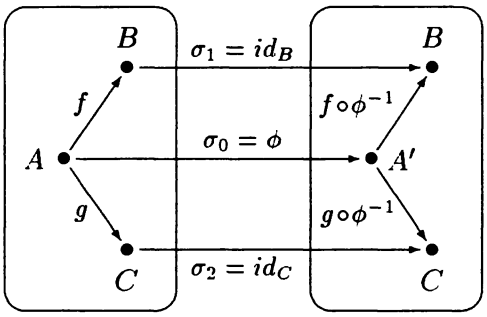
$$\overline{\pi}' = \mathsf{Subst\_Obj\_Iso}(\overline{\pi}, 0, A', \phi).$$



graphe π^Φ diagramme $\overline{\pi}$ diagramme $\overline{\pi}'$

L'isomorphisme $\overline{\sigma} : \overline{\pi} \to \overline{\pi}'$ est composé du morphisme de graphes $\sigma^\Phi = id_{\pi^\Phi}$ et de la transformation naturelle généralisée $\sigma : \pi \rightsquigarrow \pi' \circ \sigma^\Phi$ définie par

$$\sigma_0 = \phi, \ \sigma_1 = id_B, \ \sigma_2 = id_C.$$



isomorphisme $\overline{\sigma} : \overline{\pi} \to \overline{\pi}'$

5.1.2 Suppression d'un arc

Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Intuitivement, *supprimer l'arc* $a_0 : n_0 \to n_0'$ du diagramme $\overline{\delta}$ consiste à retirer l'arc a_0 du graphe δ^Φ et à conserver les mêmes étiquettes pour les arcs et les nœuds restants.

Définition 5.4 (Suppr_Arc) Soit un diagramme $\overline{\delta}$ et un arc $a_0 : n_0 \to n_0' \in \mathrm{Arc}(\delta^\Phi)$. Le diagramme

$$\overline{\beta} = \mathsf{Suppr\_Arc}(\overline{\delta}, a_0)$$

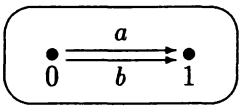
est défini de la façon suivante.

- Le graphe β^Φ est le graphe δ^Φ auquel on retire l'arc a_0.
  - $\mathrm{Nœud}(\beta^\Phi) = \mathrm{Nœud}(\delta^\Phi)$;
  - $\mathrm{Arc}(\beta^\Phi) = \mathrm{Arc}(\delta^\Phi) - \{a_0\}$.

- Le foncteur $\beta : \beta^\Phi \to \mathcal{C}_0$ est la restriction de $\delta : \delta^\Phi \to \mathcal{C}_0$ à β^Φ.
  - $\forall n \in \text{Nœud}(\beta^\Phi),\ \beta(n) = \delta(n)$;
  - $\forall a : n \to n' \in \text{Arc}(\beta^\Phi),\ \beta(a) = \delta(a)$.

Dans le cas général, la transformation Suppr_Arc n'est évidemment pas stable par isomorphisme.

Exemple 5.2 Considérons le diagramme $\overline{\kappa}$, construit sur le graphe κ^Φ.



graphe κ^Φ

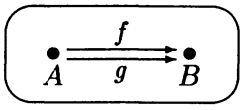
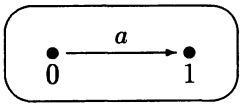


diagramme $\overline{\kappa}$

Posons $\overline{\kappa}' = \text{Suppr\_Arc}(\overline{\kappa}, b)$.



graphe κ'^Φ

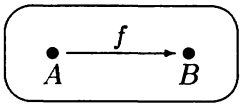


diagramme $\overline{\kappa}'$

Dans le cas général, les diagrammes $\overline{\kappa}$ et $\overline{\kappa}'$ ne sont pas isomorphes dans la catégorie $\text{diagr}(\mathcal{C}_0)$.

Par contre, la suppression de certains arcs conduit à des diagrammes isomorphes. En particulier, on peut d'une part supprimer les *boucles identités* et d'autre part l'un des arcs d'un *doublet.*

Une *boucle* sur un graphe δ^Φ est un arc $a : n \to n$ de δ^Φ qui a le même nœud pour source et but.

Définition 5.5 (Boucle) Soit un graphe δ^Φ. Un arc a de δ^Φ est une *boucle* si et seulement si $\text{Source}(a) = \text{But}(a)$.

Une *boucle identité* sur un diagramme $\overline{\delta}$ est une boucle de δ^Φ étiquetée par une flèche identité.

Définition 5.6 (Boucle identité) Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Une *boucle identité* sur $\overline{\delta}$ est une boucle $a : n \to n$ de δ^Φ telle que $\delta(a) = id_{\delta(n)}$.

Proposition 5.2 *Soit un diagramme* $\overline{\delta}$ *et une boucle identité* $a_0 : n_0 \to n_0$ *sur* $\overline{\delta}$. *Alors, dans la catégorie* $\text{diagr}(\mathcal{C}_0)$,

$$\text{Suppr\_Arc}(\overline{\delta}, a_0) \cong \overline{\delta}.$$

Preuve. Considérons un diagramme $\overline{\delta}$ et une boucle identité $a_0 : n_0 \to n_0$ de $\overline{\delta}$. Soit $\overline{\beta} = \mathsf{Suppr\_Arc}(\overline{\delta}, a_0)$. On construit deux flèches de DIAGR($\mathcal{C}_0$), $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_{\beta}}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\delta}}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\sigma^{\Phi} : \delta^{\Phi} \to \beta^{\Phi}$.
    - $\forall n \in \text{Nœud}(\delta^{\Phi})$, $\sigma^{\Phi}(n) = n$;
    - $\forall a \in \text{Arc}(\delta^{\Phi})$, $a \neq a_0$, $\sigma^{\Phi}(a) = a$;
    - $\sigma^{\Phi}(a_0) = 0_{n_0}$ (zigzag nul de source et but n_0).
  - Transformation naturelle généralisée $\sigma : \delta \dot{\rightsquigarrow} \beta \circ \sigma^{\Phi}$.
    - $\forall n \in \text{Nœud}(\delta^{\Phi})$, $\sigma_n = id_{\delta(n)}$.

    Cela définit bien une transformation naturelle généralisée car $\delta(a_0) = id_{\delta(n_0)}$.
- On définit $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\tau^{\Phi} : \beta^{\Phi} \to \delta^{\Phi}$.
    - $\forall n \in \text{Nœud}(\beta^{\Phi})$, $\tau^{\Phi}(n) = n$;
    - $\forall a \in \text{Arc}(\beta^{\Phi})$, $\tau^{\Phi}(a) = a$.
  - Transformation naturelle généralisée $\tau : \beta \dot{\rightsquigarrow} \delta \circ \tau^{\Phi}$.
    - $\forall n \in \text{Nœud}(\beta^{\Phi})$, $\tau_n = id_{\beta(n)}$.
- On a $\overline{\sigma} \circ \overline{\tau} = \overline{id_{\beta}}$.
- On vérifie également que $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\delta}}$.

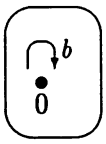
Par conséquent, les diagrammes $\overline{\delta}$ et $\overline{\beta}$ sont isomorphes dans la catégorie diagr($\mathcal{C}_0$). □

Exemple 5.3 Soit A un objet de $\mathcal{C}_0$. Considérons le diagramme $I_{\mathcal{C}_0}(A)$, construit sur le graphe 1^{Φ}. Rappelons que le graphe 1^{Φ} contient un seul nœud appelé $*$.

graphe 1^{Φ}

diagramme $I_{\mathcal{C}_0}(A)$

Considérons le graphe ρ^{Φ} qui contient un nœud appelé 0 et un arc $b : 0 \to 0$. Étant donné un objet A et une flèche $f : A \to A$ de $\mathcal{C}_0$, on note $\overline{\rho}(A, f)$ le diagramme construit sur le graphe ρ^{Φ}, qui associe au nœud 0 l'objet A et à l'arc b la flèche f. Nous considérons ici le diagramme $\overline{\rho}(A, id_A)$.



graphe ρ^{Φ}

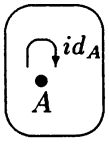


diagramme $\overline{\rho}(A, id_A)$

Les diagrammes $I_{\mathcal{C}_0}(A)$ et $\overline{\rho}(A, id_A)$ sont isomorphes, car

$$I_{\mathcal{C}_0}(A) = \mathsf{Suppr\_Arc}(\overline{\rho}(A, id_A), b)$$

et b est une boucle identité sur $\overline{\rho}(A, id_A)$.

Un doublet sur un diagramme $\overline{\delta}$ est un couple d'arcs qui ont tous les deux la même source et le même but, et étiquetés par la même flèche de $\mathcal{C}_0$.

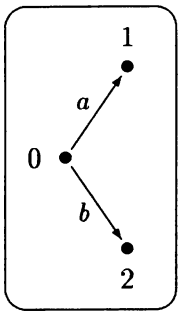
Définition 5.7 (Doublet) Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Un *doublet* sur $\overline{\delta}$ est un couple d'arcs (a_0, a_1) tel que

$$\mathrm{Source}(a_0) = \mathrm{Source}(a_1),\ \mathrm{But}(a_0) = \mathrm{But}(a_1) \text{ et } \delta(a_0) = \delta(a_1).$$

Exemple 5.4

- Soit deux objets A et B et une flèche $f : A \to B$ de $\mathcal{C}_0$. Considérons le diagramme $\overline{\pi_1}$, construit sur le graphe π^Φ, défini par

  $\pi(0) = A$;
  $\pi(1) = \pi(2) = B$;
  $\pi(a) = \pi(b) = f$.



graphe π^Φ

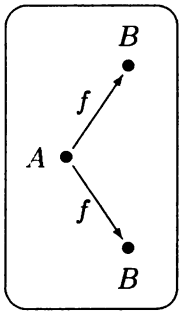
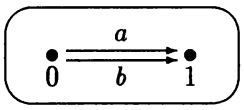


diagramme $\overline{\pi_1}$

  Le couple (a, b) n'est pas un doublet sur $\overline{\pi_1}$, bien que les arcs a et b soient étiquetés par la même flèche f de $\mathcal{C}_0$, car ces arcs n'ont pas le même but.

- Considérons le diagramme $\overline{\kappa_1}$, construit sur le graphe κ^Φ.



graphe κ^Φ

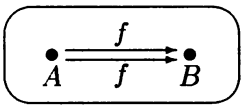


diagramme $\overline{\kappa_1}$

  Le couple (a, b) est un doublet de $\overline{\kappa_1}$.

Proposition 5.3 *Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$ et un doublet (a_0, a_1) de $\overline{\delta}$. Alors, dans la catégorie* $\mathrm{diagr}(\mathcal{C}_0)$,

$$\mathsf{Suppr\_Arc}(\overline{\delta}, a_0) \cong \overline{\delta}.$$

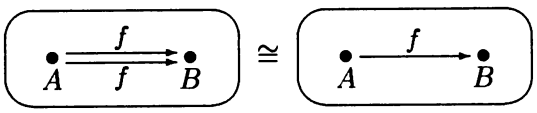
Preuve. Soit un diagramme $\overline{\delta}$ et un doublet (a_0, a_1) de $\overline{\delta}$. Soit $\overline{\beta} = \mathsf{Suppr\_Arc}(\overline{\delta}, a_0)$. On construit deux flèches de $\mathrm{DIAGR}(\mathcal{C}_0)$, $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_{\beta}}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\delta}}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\sigma^{\Phi} : \delta^{\Phi} \to \beta^{\Phi}$.
    - $\forall n \in \mathrm{Nœud}(\delta^{\Phi})$, $\sigma^{\Phi}(n) = n$;
    - $\forall a \in \mathrm{Arc}(\delta^{\Phi})$, $a \neq a_0$, $\sigma^{\Phi}(a) = a$;
    - $\sigma^{\Phi}(a_0) = a_1$.
  - Transformation naturelle généralisée $\sigma : \delta \rightsquigarrow \beta \circ \sigma^{\Phi}$.
    - $\forall n \in \mathrm{Nœud}(\delta^{\Phi})$, $\sigma_n = id_{\delta(n)}$.

    Cela définit bien une transformation naturelle généralisée. En effet, on a $\delta(a_0) = \delta(a_1)$.
- On définit $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\tau^{\Phi} : \beta^{\Phi} \to \delta^{\Phi}$.
    - $\forall n \in \mathrm{Nœud}(\beta^{\Phi})$, $\tau^{\Phi}(n) = n$;
    - $\forall a \in \mathrm{Arc}(\beta^{\Phi})$, $\tau^{\Phi}(a) = a$.
  - Transformation naturelle généralisée $\tau : \beta \rightsquigarrow \delta \circ \tau^{\Phi}$.
    - $\forall n \in \mathrm{Nœud}(\beta^{\Phi})$, $\tau_n = id_{\beta(n)}$.
- On a $\overline{\sigma} \circ \overline{\tau} = \overline{id_{\beta}}$.
- On vérifie également que $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\delta}}$.

Par conséquent, les diagrammes $\overline{\delta}$ et $\overline{\beta}$ sont isomorphes dans la catégorie $\mathrm{diagr}(\mathcal{C}_0)$. □

Exemple 5.5 Les diagrammes suivants sont isomorphes dans $\mathrm{diagr}(\mathcal{C}_0)$.



5.1.3 Ajout d'un arc étiqueté

Étant donné un diagramme $\overline{\delta}$, l'opération d'*ajout d'arc étiqueté* consiste à ajouter un arc au diagramme sous-jacent δ^{Φ} et à étiqueter cet arc par une flèche de $\mathcal{C}_0$.

Définition 5.8 (Ajout_Arc) Soit un diagramme $\overline{\delta}$, deux nœuds n_0 et n_1 de δ^{Φ} et une flèche $f : \delta(n_0) \to \delta(n_1)$ de $\mathcal{C}_0$. L'ajout d'un arc de n_0 vers n_1 étiqueté par f dans le diagramme $\overline{\delta}$ est le diagramme

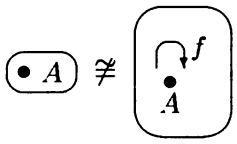
$$\overline{\beta} = \mathsf{Ajout\_Arc}(\overline{\delta}, n_0, n_1, f)$$

défini de la façon suivante.

- Le graphe β^Φ est le graphe δ^Φ auquel on ajoute un nouvel arc $a_x : n_0 \to n_1$.
  - $\text{Nœud}(\beta^\Phi) = \text{Nœud}(\delta^\Phi)$;
  - $\text{Arc}(\beta^\Phi) = \text{Arc}(\delta^\Phi) \cup \{a_x\}$, où $a_x \notin \text{Arc}(\delta^\Phi)$, avec $\text{Source}(a_x) = n_0$ et $\text{But}(a_x) = n_1$.
- Le foncteur $\beta : \beta^\Phi \to \mathcal{C}_0$ étend le foncteur δ en associant à l'arc a_x la flèche f.
  - Action sur les nœuds.
    - $\forall n \in \text{Nœud}(\beta^\Phi),\ \beta(n) = \delta(n)$.
  - Action sur les arcs.
    - $\forall a \in \text{Arc}(\beta^\Phi),\ a \neq a_x,\ \beta(a) = \delta(a)$;
    - $\beta(a_x) = f$.

La transformation Ajout_Arc n'est évidemment pas stable par isomorphisme dans le cas général.

Exemple 5.6 Soit un objet A et une flèche $f : A \to A$ dans $\mathcal{C}_0$. Dans le cas général, les diagrammes $I_{\mathcal{C}_0}(A)$ et $\overline{p}(A, f)$ ne sont pas isomorphes dans $\text{diagr}(\mathcal{C}_0)$.



Proposition 5.4 *(Ajout de compositions) Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Soit deux arcs $a_0 : n_0 \to n_1$ et $a_1 : n_1 \to n_2$ de δ^Φ. Alors, dans la catégorie* $\text{diagr}(\mathcal{C}_0)$,

$$\text{Ajout\_Arc}(\overline{\delta}, n_0, n_2, \delta(a_1) \circ \delta(a_0)) \cong \overline{\delta}.$$

Preuve. Soit un diagramme $\overline{\delta}$ et deux arcs $a_0 : n_0 \to n_1$ et $a_1 : n_1 \to n_2$ de δ^Φ. Posons $\overline{\beta} = \text{Ajout\_Arc}(\overline{\delta}, n_0, n_2, \delta(a_1) \circ \delta(a_0))$. On construit deux flèches de $\text{DIAGR}(\mathcal{C}_0)$, $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id}_{\overline{\beta}}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id}_{\overline{\delta}}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\sigma^\Phi : \delta^\Phi \to \beta^\Phi$.
    - $\forall n \in \text{Nœud}(\delta^\Phi),\ \sigma^\Phi(n) = n$;
    - $\forall a \in \text{Arc}(\delta^\Phi),\ \sigma^\Phi(a) = a$.
  - Transformation naturelle généralisée $\sigma : \delta \rightsquigarrow \beta \circ \sigma^\Phi$.
    - $\forall n \in \text{Nœud}(\delta^\Phi),\ \sigma_n = id_{\delta(n)}$.
- On définit $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\tau^\Phi : \beta^\Phi \to \delta^\Phi$.
    - $\forall n \in \text{Nœud}(\beta^\Phi),\ \tau^\Phi(n) = n$;

· $\forall a \in \mathrm{Arc}(\beta^\Phi),\ a \neq a_x,\ \tau^\Phi(a) = a$;

· $\tau^\Phi(a_x) = n_0 \xrightarrow{a_0} n_1 \xrightarrow{a_1} n_2$ (zigzag de longueur de 2).

– Transformation naturelle généralisée $\tau : \beta \rightsquigarrow \delta \circ \tau^\Phi$.

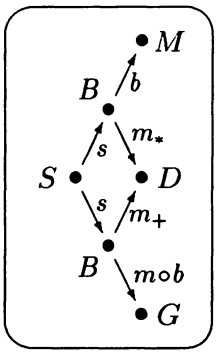
· $\forall n \in \mathrm{Nœud}(\beta^\Phi),\ \tau_n = id_{\beta(n)}$.

Cela définit bien une transformation naturelle généralisée. En effet, on a $\beta(a_x) = \delta(a_1) \circ \delta(a_0)$.

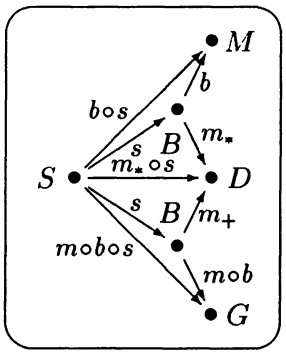
- On a $\overline{\tau} \circ \overline{\sigma} = \overline{id_{\overline{\delta}}}$.
- On vérifie également que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_{\overline{\beta}}}$.

□

Exemple 5.7 Considérons le diagramme $\overline{\delta_4}$ sur $\mathcal{C}_0$. Nous avons vu une première fois ce diagramme dans la partie I, chapitre 1, page 45, et une seconde fois dans le chapitre 4, page 26. Nous pouvons ajouter trois arcs étiquetés dans $\overline{\delta_4}$: un arc étiqueté par $b \circ s$ entre (les nœuds étiquetés par) S et M, un arc étiqueté par $m_* \circ s$ entre S et D et un arc étiqueté par $m \circ b \circ s$ entre S et G. Nous obtenons ainsi un diagramme $\overline{\delta'}$.
Nous n'avons pas ajouté d'arc étiqueté par $m_+ \circ s$ entre S et D car il existe déjà un arc étiqueté par $m_* \circ s$ entre ces deux nœuds et, dans la catégorie $\mathcal{C}_0$, on a l'égalité $m_+ \circ s = m_* \circ s$.



$\overline{\delta_4}$



$\overline{\delta'}$

Proposition 5.5 *(Ajout de factorisations à droite) Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Soit deux arcs $a_1 : n_1 \to n_0$ et $a_2 : n_2 \to n_0$ de δ^Φ. Soit une flèche $h : \delta(n_1) \to \delta(n_2)$ de $\mathcal{C}_0$ telle que $\delta(a_2) \circ h = \delta(a_1)$. Alors, dans la catégorie* $\mathrm{diagr}(\mathcal{C}_0)$,

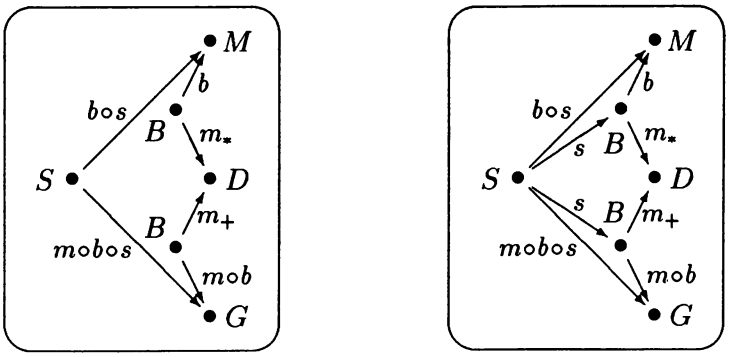
$$\mathsf{Ajout\_Arc}(\overline{\delta}, n_1, n_2, h) \cong \overline{\delta}.$$

Preuve. La preuve est similaire à la preuve de la proposition 5.4. On pose

$$\tau^\Phi(a_x) = n_1 \xrightarrow{a_1} n_0 \xleftarrow{a_2} n_2.$$

Cela définit bien une transformation naturelle généralisée $\tau : \beta \rightsquigarrow \delta \circ \tau^\Phi$. En effet, $\delta(a_2) \circ h = \delta(a_1)$. □

Exemple 5.8 Considérons le diagramme $\overline{\delta_3}$, vu une première fois dans la partie I, chapitre 1, page 45, et revu dans le chapitre 4, page 23. Nous obtenons le diagramme $\overline{\delta'_3}$ à partir du diagramme $\overline{\delta_3}$ en ajoutant deux factorisations à droite.



$\overline{\delta_3}$ $\overline{\delta'_3}$

5.1.4 Contraction d'un arc identité

Soit un diagramme $\overline{\delta}$. Un *arc identité* du diagramme $\overline{\delta}$ est un arc de δ^Φ étiqueté par une flèche identité.

Définition 5.9 (Arc identité) Soit un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$. Un arc $a_0 : n_0 \to n_1$ du graphe δ^Φ est un *arc identité* si et seulement si $\delta(n_0) = \delta(n_1)$ et $\delta(a_0) = id_{\delta(n_0)}$.

Intuitivement, *contracter un arc identité* $a_0 : n_0 \to n_1$ dans un diagramme $\overline{\delta}$ consiste à fusionner les nœuds n_0 et n_1 et à supprimer l'arc a_0.

Définition 5.10 (Contract_Id) Soit un diagramme $\overline{\delta}$ et un arc identité $a_0 : n_0 \to n_1$ de $\overline{\delta}$, tel que les nœuds n_0 et n_1 sont distincts. Le diagramme

$$\overline{\beta} = \mathsf{Contract\_Id}(\overline{\delta}, a_0)$$

est défini de la façon suivante.

- Graphe β^Φ.
  - $\text{Nœud}(\beta^\Phi) = \text{Nœud}(\delta^\Phi) - \{n_1\}$
  - L'ensemble $\text{Arc}(\beta^\Phi)$ est défini à partir de l'ensemble $\text{Arc}(\delta^\Phi)$ de la façon suivante. Soit un arc $a : n \to n'$ de δ^Φ.
    - $n \neq n_1$ et $n' \neq n_1 \Rightarrow a : n \to n' \in \text{Arc}(\beta^\Phi)$;
    - $n = n_1$ et $n' \neq n_1 \Rightarrow a : n_0 \to n' \in \text{Arc}(\beta^\Phi)$;
    - $n \neq n_1$ et $n' = n_1$ et $a \neq a_0 \Rightarrow a : n \to n_0 \in \text{Arc}(\beta^\Phi)$;
    - $n = n_1$ et $n' = n_1 \Rightarrow a : n_0 \to n_0 \in \text{Arc}(\beta^\Phi)$.
- Foncteur $\beta : \beta^\Phi \to \mathcal{C}_0$.
  - $\forall n \in \text{Nœud}(\beta^\Phi),\ \beta(n) = \delta(n)$;

- $\forall a \in \text{Arc}(\beta^\Phi),\ \beta(a) = \delta(a)$.

Nous avons supposé que les nœuds n_0 et n_1 sont distincts, car si $n_0 = n_1$, cela n'a pas de sens de fusionner ces deux nœuds. De plus, si $n_0 = n_1$, alors a_0 est *une boucle identité*, qui peut être supprimée par l'opération Suppr_Arc (cf. proposition 5.2).

Proposition 5.6 *La transformation* Contract_Id *est stable par isomorphisme.*

Preuve. Soit un diagramme $\overline{\delta}$ et un arc identité $a_0 : n_0 \to n_1$ de $\overline{\delta}$, tel que $n_0 \neq n_1$. Posons $\overline{\beta} = \text{Contract\_Id}(\overline{\delta}, a_0)$. On construit deux flèches de $\text{DIAGR}(\mathcal{C}_0)$, $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_\beta}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_\delta}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\sigma^\Phi : \delta^\Phi \to \beta^\Phi$.
    - $\forall n \in \text{Nœud}(\delta^\Phi),\ n \neq n_1,\ \sigma^\Phi(n) = n$;
    - $\sigma^\Phi(n_1) = n_0$;
    - $\forall a \in \text{Arc}(\delta^\Phi),\ a \neq a_0,\ \sigma^\Phi(a) = a$;
    - $\sigma^\Phi(a_0) = 0_{n_0}$ (zigzag nul ayant pour source et but le nœud n_0 de β^Φ).
  - Transformation naturelle généralisée $\sigma : \delta \dot{\rightsquigarrow} \beta \circ \sigma^\Phi$.
    - $\forall n \in \text{Nœud}(\delta^\Phi),\ \sigma_n = id_{\delta(n)}$.

    Cela définit bien une transformation naturelle généralisée car $\delta(a_0) = id_{\delta(n_0)}$.

- On définit $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\tau^\Phi : \beta^\Phi \to \delta^\Phi$.
    - $\forall n \in \text{Nœud}(\beta^\Phi),\ \tau^\Phi(n) = n$;
    - $\forall a \in \text{Arc}(\beta^\Phi),\ \tau^\Phi(a) = a$.
  - Transformation naturelle généralisée $\tau : \beta \dot{\rightsquigarrow} \delta \circ \tau^\Phi$.
    - $\forall n \in \text{Nœud}(\beta^\Phi),\ \tau_n = id_{\beta(n)}$.

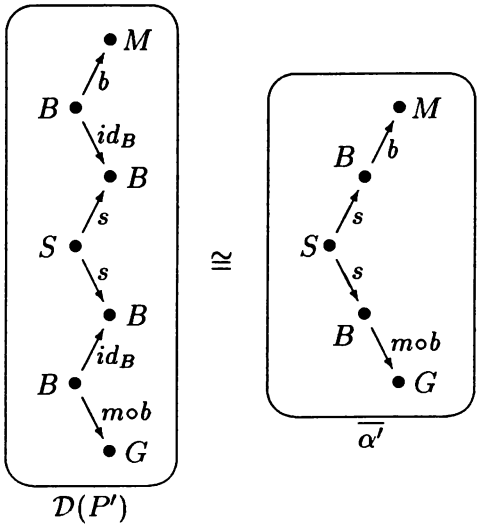
- On a $\overline{\sigma} \circ \overline{\tau} = \overline{id_\beta}$.

- On vérifie également que $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_\delta}$.

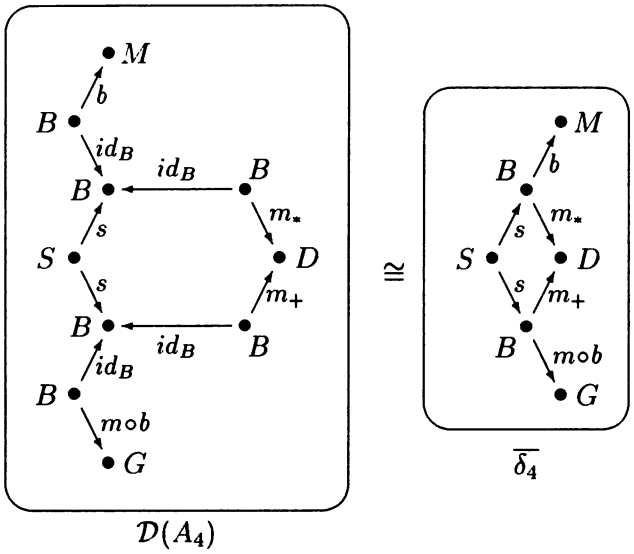
□

Exemple 5.9

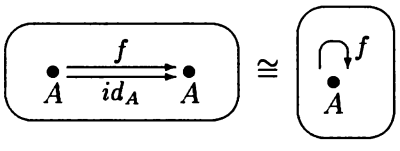
1. Les diagrammes $\mathcal{D}(P')$ et $\overline{\alpha'}$ (cf. chapitre 4, page 25) sont isomorphes dans la catégorie diagr($\mathcal{C}_0$).



2. Les diagrammes $\mathcal{D}(A_4)$ et $\overline{\delta_4}$ (cf. chapitre 4, page 26) sont isomorphes dans la catégorie diagr($\mathcal{C}_0$).



3. Les diagrammes suivants sont isomorphes dans la catégorie diagr($\mathcal{C}_0$).



Cet exemple montre qu'à partir d'un diagramme dont le graphe sous-jacent n'est pas cyclique, la contraction d'un arc identité peut produire un diagramme dont le graphe sous-jacent est cyclique.

5.2 Catégorie $\mathcal{C}_0$ finie et sans cycle

Dans cette partie, nous nous intéressons à la détection d'isomorphismes lorsque la catégorie $\mathcal{C}_0$ est finie et sans cycle. Nous commençons par définir un *cycle* dans une catégorie.

Définition 5.11 (Cycle) Étant donné une catégorie $\mathcal{C}$, un *cycle* dans $\mathcal{C}$ est une flèche $f : A \to A$ ayant même domaine et codomaine A et telle que $f \neq id_A$.
Nous dirons qu'une catégorie $\mathcal{C}$ *ne comporte pas de cycle* si et seulement si pour toute flèche $f : A \to A$ de $\mathcal{C}$, $f = id_A$.

5.2.1 Hypothèses

Nous supposons que la catégorie de base $\mathcal{C}_0$ est finie et ne comporte pas de cycle. Cette hypothèse est compatible avec le langage de spécification LPG, puisque d'une part on ne peut déclarer qu'un nombre fini de spécifications et morphismes de spécifications, et d'autre part, la déclaration d'un morphisme de spécifications ayant la même spécification pour domaine et codomaine est syntaxiquement interdite.

D'autre part, nous supposons qu'*il n'existe pas d'isomorphisme entre objets* dans la catégorie de base $\mathcal{C}_0$. Cette hypothèse est justifiée par la transformation de substitution par objet isomorphe (proposition 5.1). En effet, supposons qu'on a, au départ, une catégorie $\mathcal{X}_0$ qui ne possède pas cette propriété. Pour chaque classe d'isomophisme entre objets de $\mathcal{X}_0$, on choisit un représentant. On considère ensuite la sous-catégorie pleine $\mathcal{C}_0$ de $\mathcal{X}_0$ ayant pour objets ces représentants. La catégorie $\mathcal{C}_0$ satisfait alors la propriété suivante :

$$\text{pour tout couple d'objets } (A, B) \text{ de } \mathcal{C}_0,\ A \cong B \Rightarrow A = B.$$

Tout diagramme sur $\mathcal{X}_0$ peut être transformé en un diagramme sur $\mathcal{C}_0$ en utilisant la proposition 5.1.

Nous faisons également l'hypothèse qu'on sait détecter une égalité entre deux morphismes de spécifications de base, c'est-à-dire que l'égalité est décidable dans $\mathcal{C}_0$. Enfin nous supposons que toutes les factorisations à droite sont connues.

Hypothèses

1. $\mathcal{C}_0$ est finie.
2. $\mathcal{C}_0$ ne comporte pas de cycle.
3. Pour tout couple d'objets (A, B) de $\mathcal{C}_0$, $A \cong B \Rightarrow A = B$.
4. L'égalité sur $\mathcal{C}_0$ est décidable. Soit deux flèches $f, g : A \to B$ de $\mathcal{C}_0$. On sait si $f = g$ ou $f \neq g$.
5. On sait effectuer des factorisations à droite dans $\mathcal{C}_0$. Étant donné deux flèches $f : A \to B$ et $g : C \to B$, on connaît l'ensemble des flèches $h : A \to C$ telles que $f = g \circ h$.

Remarque 5.1 Les hypothèses 1 et 2 sont distinctes. En effet, il existe des catégories infinies qui ne comportent pas de cycle, et il existe des catégories finies qui comportent des cycles.

La combinaison des hypothèses 2 et 3 est très forte, comme le montre le lemme suivant.

Lemme 5.1 *Soit deux objets A et B ainsi que deux flèches $f : A \to B$ et $g : B \to A$ dans $\mathcal{C}_0$. Alors, $A = B$ et $f = g = id_A$.*

Preuve. D'après l'hypothèse 2, $g \circ f = id_A$ et $f \circ g = id_B$, d'où $A \cong B$. D'après l'hypothèse 3, on a donc $A = B$. On en déduit que f et g sont des flèches de A vers A. Par conséquent, en utilisant de nouveau l'hypothèse 2, on obtient $f = g = id_A$. □

5.2.2 Fonction de complétion

Dans ce paragraphe, nous détaillons la fonction de *complétion* d'un diagramme. Compléter un diagramme $\overline{\alpha}$ sur $\mathcal{C}_0$ consiste à contracter les arcs identités, supprimer les boucles identités, supprimer les doublets, ajouter les compositions d'arcs et les factorisations à droite.

Fonction Complétion ($\overline{\alpha}$: diagramme): diagramme

$\overline{\delta}$, $\overline{\delta_0}$: diagramme;

Début

$\overline{\delta} := \overline{\alpha}$;

Répéter

$\overline{\delta_0} := \overline{\delta}$;

Pour tout arc identité a de $\overline{\delta}$: (1)

$\overline{\delta} :=$ Contract_Id$(\overline{\delta}, a)$;

Pour toute boucle identité a de $\overline{\delta}$: (2)

$\overline{\delta} :=$ Suppr_Arc$(\overline{\delta}, a)$;

Pour tout doublet (a_0, a_1) de $\overline{\delta}$: (3)

$\overline{\delta} :=$ Suppr_Arc$(\overline{\delta}, a_0)$;

Pour tout couple d'arcs $(a_0 : n_0 \to n_1,\ a_1 : n_1 \to n_2)$ de $\overline{\delta}$ tel que (4)
il n'existe pas d'arc $a : n_0 \to n_2$ étiqueté par $\delta(a_1) \circ \delta(a_0)$ dans $\overline{\delta}$,

$\overline{\delta} :=$ Ajout_Arc$(\overline{\delta}, n_0, n_2, \delta(a_1) \circ \delta(a_0))$;

Pour tout couple d'arcs $(a_1 : n_1 \to n_0,\ a_2 : n_2 \to n_0)$ de $\overline{\delta}$ tel que (5)

- il existe une flèche $h : \delta(n_1) \to \delta(n_2)$ de $\mathcal{C}_0$ vérifiant $\delta(a_2) \circ h = \delta(a_1)$,
- il n'existe pas d'arc $a : n_1 \to n_2$ étiqueté par h dans $\overline{\delta}$,

$\overline{\delta} :=$ Ajout_Arc$(\overline{\delta}, n_1, n_2, h)$;

Jusqu'à $\overline{\delta} = \overline{\delta_0}$;

Résultat : $\overline{\delta}$;

Fin Complétion

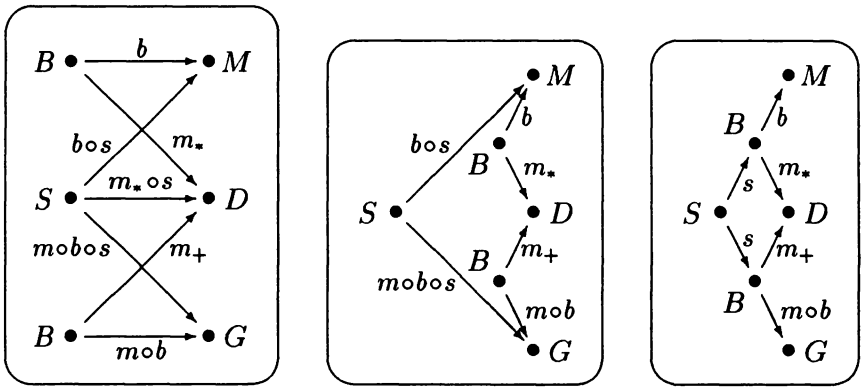
Proposition 5.7 *La fonction de complétion s'arrête.*

Preuve. On appelle *itération* le fait d'exécuter successivement les étapes 1 à 5. D'après la condition d'arrêt, on effectue une itération tant que le diagramme a été modifié par l'itération précédente. Chaque étape de l'itération termine : les étapes 1, 2 et 3 parce qu'elles suppriment un arc dans le graphe, et les étapes 4 et 5 parce que la catégorie $\mathcal{C}_0$ est finie et qu'on n'ajoute pas une nouvelle composition ou factorisation à droite si l'arc correspondant appartient déjà au diagramme.
On ne peut effectuer qu'un nombre fini d'itérations. En effet, seule la contraction d'un arc identité (étape 1) peut permettre ensuite d'ajouter des nouvelles compositions d'arcs ou des nouvelles factorisations à droite. Comme la contraction supprime un nœud dans le graphe, et comme aucune transformation n'ajoute de nœud dans le graphe, on ne peut effectuer qu'un nombre fini d'itérations. □

Proposition 5.8 *La complétion est une transformation stable par isomorphisme. Pour tout diagramme $\overline{\delta}$ sur $\mathcal{C}_0$, $\overline{\delta} \cong$ Complétion$(\overline{\delta})$ dans* diagr$(\mathcal{C}_0)$.

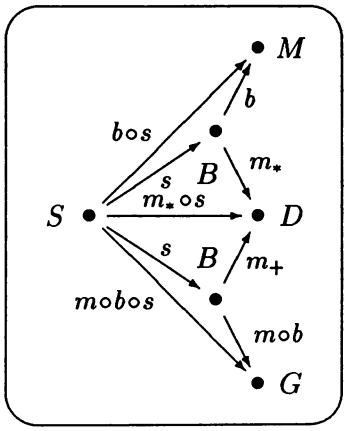
Preuve. Toutes les transformations appliquées sur le diagramme $\overline{\delta}$ par la fonction de complétion sont stables par isomorphisme, d'après les propositions 5.2, 5.3, 5.4, 5.5, et 5.6. □

Exemple 5.10 Considérons les diagrammes $\overline{\delta_1}$, $\overline{\delta_3}$ et $\overline{\delta_4}$.



$\overline{\delta_1}$ $\overline{\delta_3}$ $\overline{\delta_4}$

La complétion de ces trois diagrammes produit le même diagramme $\overline{\delta'}$:
Complétion($\overline{\delta_1}$) = Complétion($\overline{\delta_3}$) = Complétion($\overline{\delta_4}$) = $\overline{\delta'}$.

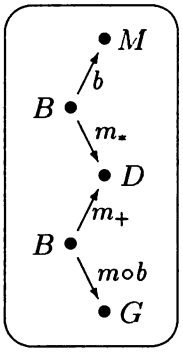


$\overline{\delta'}$

Un diagramme est *complet* si celui-ci est égal à sa complétion.

Définition 5.12 (Diagramme complet) Un diagramme $\overline{\delta}$ sur $\mathcal{C}_0$ est *complet* si et seulement si $\overline{\delta}$ = Complétion($\overline{\delta}$).

Exemple 5.11 Le diagramme $\overline{\delta_2}$ est complet : $\overline{\delta_2}$ = Complétion($\overline{\delta_2}$).



$\overline{\delta_2}$

Lemme 5.2 *Soit un diagramme complet $\overline{\delta}$ sur $\mathcal{C}_0$. Alors il n'existe aucune boucle dans le graphe δ^Φ.*

Preuve. Supposons par l'absurde qu'on a une boucle, c'est-à-dire un arc $a : n \to n$ dans δ^Φ. On a soit $\delta(a) = id_{\delta(n)}$, soit $\delta(a) \neq id_{\delta(n)}$. La supposition $\delta(a) = id_{\delta(n)}$ est en contradiction avec l'hypothèse que $\overline{\delta}$ est complet. La supposition $\delta(a) \neq id_{\delta(n)}$ est en contradiction avec l'hypothèse que $\mathcal{C}_0$ ne comporte pas de cycle. Par conséquent, il n'existe pas de boucle dans le graphe δ^Φ. □

Lemme 5.3 *Soit un diagramme complet $\overline{\delta}$ sur $\mathcal{C}_0$. Soit deux nœuds n_1, n_2 et un arc $a : n_1 \to n_2$ dans le graphe δ^Φ. Alors, il n'existe aucun arc de n_2 vers n_1 dans δ^Φ.*

Preuve. S'il existe un arc $a' : n_2 \to n_1$ dans δ^Φ, alors comme $\overline{\delta}$ est complet, il existe un arc $a_1 : n_1 \to n_1$ étiqueté par la flèche $\delta(a') \circ \delta(a)$, ce qui est impossible d'après le lemme 5.2. □

5.2.3 Zigzags élémentaires et liens

La complétion d'un diagramme permet d'obtenir une forme un peu plus "canonique" pour les diagrammes. Cependant, deux diagrammes complets peuvent être isomorphes dans diagr($\mathcal{C}_0$) sans être identiques. Par exemple, les diagrammes $\overline{\delta_2}$ et $\overline{\delta'}$ sont complets et isomorphes, mais ne sont pas identiques. Pour détecter cet isomorphisme, nous devons introduire la notion de *zigzag élémentaire.*

Définition 5.13 (Zigzag élémentaire) Un *zigzag élémentaire* sur un graphe δ^Φ est un zigzag de longueur 2 de la forme

$$m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2$$

tel que les arcs a_0 et a_1 sont distincts.

Nous avons également besoin d'une relation d'*inclusion* sur les zigzags. Intuitivement, le zigzag Z est inclus dans le zigzag Z' si Z est un "sous-zigzag" de Z'.

Définition 5.14 (Inclusion de zigzags) Soit un graphe δ^Φ. Considérons deux zigzags Z et Z' sur δ^Φ. Posons $Z = (k, Z_N, Z_A)$ et $Z' = (k', Z'_N, Z'_A)$, avec

$$\begin{array}{l} Z_N = (n_0, n_1, \ldots, n_k) \quad \text{et} \quad Z_A = (a_0, a_1, \ldots, a_{k-1}) ; \\ Z'_N = (n'_0, n'_1, \ldots, n'_{k'}) \quad \text{et} \quad Z'_A = (a'_0, a'_1, \ldots, a'_{k'-1}). \end{array}$$

On dit que le zigzag Z est *inclus* dans le zigzag Z', et on note $Z \subseteq Z'$ si et seulement si il existe un entier j, $0 \leq j \leq k' - k$, tel que

- $\forall i,\ 0 \leq i \leq k,\ n_i = n'_{i+j}$;
- $\forall i,\ 0 \leq i \leq k-1,\ a_i = a'_{i+j}$.

Remarquons que si $Z \subseteq Z'$, alors nécessairement $k \leq k'$.

Soit un diagramme complet $\overline{\delta}$. Les zigzags élémentaires sur le *diagramme* $\overline{\delta}$ sont les zigzags élémentaires sur le *graphe* δ^Φ. Nous définissons maintenant une relation d'ordre sur les zigzags élémentaires d'un diagramme complet.

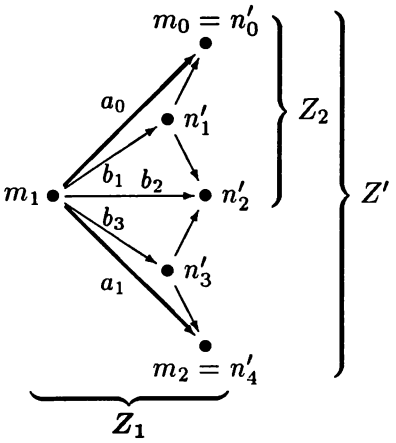
Définition 5.15 Soit un diagramme complet $\overline{\delta}$ sur $\mathcal{C}_0$. On définit une *relation d'ordre* $\leq$ *sur les zigzags élémentaires du diagramme complet* $\overline{\delta}$ de la façon suivante. Soit deux zigzags élémentaires Z_1 et Z_2 sur $\overline{\delta}$. Posons $Z_1 = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2$.

- On a $Z_1 < Z_2$ si et seulement si il existe un zigzag $Z' = (k', Z'_N, Z'_A)$, avec $Z'_N = (n'_0, n'_1, \ldots, n'_{k'})$ tel que
  - $m_0 = n'_0$ et $m_2 = n'_{k'}$;

- $Z_2 \subseteq Z'$;
- $\delta(a_0) \sim_{\overline{\delta}} \delta(a_1)$ $[Z']$;
- $\forall i,\ 0 \leq i \leq k',\ m_1 \neq n'_i$.

• On a $Z_1 \leq Z_2$ si et seulement si $Z_1 = Z_2$ ou $Z_1 < Z_2$.

Dans le dessin ci-dessous, $Z_1 < Z_2$.



Remarque 5.2 Supposons $Z_1 < Z_2$.

1. C'est le zigzag Z_1 qui paraît "le plus grand". Nous avons choisi le sens $Z_1 \leq Z_2$ car intuitivement, toute l'information sur les partages contenue dans le zigzag Z_1 est contenue dans les zigzags élémentaires de Z'.

2. Soit $\{c_i : \delta(m_1) \to \delta(n'_i)\ ;\ i \in \{0, \ldots, k'\}\}$ l'ensemble des flèches qui correspondent à la connexion $\delta(a_0) \sim_{\overline{\delta}} \delta(a_1)$ $[Z']$ (cf. définition 2.17, partie I, page 66). Alors, comme le diagramme $\overline{\delta}$ est complet, pour tout entier i, $0 \leq i \leq k'$, il existe un arc $b_i : m_1 \to n'_i$ dans δ^Φ tel que $\delta(b_i) = c_i$.

Proposition 5.9 *Soit un diagramme complet $\overline{\delta}$ sur $\mathcal{C}_0$. La relation $\leq$ sur les zigzags élémentaires sur $\overline{\delta}$ est une relation d'ordre.*

Preuve. On utilise la définition de $\leq$ ainsi que le lemme 5.3.

Réflexivité. La relation $\leq$ est réflexive par définition.

Antisymétrie. Soit deux zigzags élémentaires sur $\overline{\delta}$

$$Z_1 = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2,$$
$$Z_2 = n_0 \xleftarrow{b_0} n_1 \xrightarrow{b_1} n_2.$$

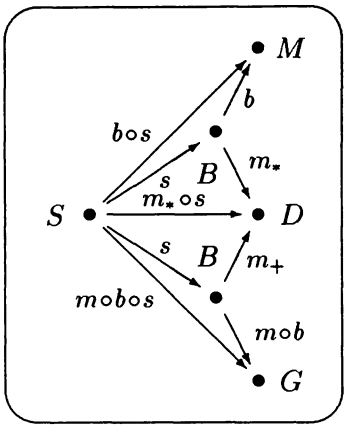
Supposons $Z_1 \leq Z_2$ et $Z_2 \leq Z_1$. Par l'absurde, si $Z_1 \neq Z_2$, alors $Z_1 < Z_2$ et $Z_2 < Z_1$. Par conséquent, on a $m_1 \neq n_1$. De plus, il existe un arc $a : m_1 \to n_1$ et un arc $a' : n_1 \to m_1$ dans δ^Φ. Cela est impossible d'après le lemme 5.3.

Transitivité. Soit trois zigzags élémentaires Z_1, Z_2 et Z_3 sur un diagramme complet $\overline{\delta}$. Supposons $Z_1 \leq Z_2$ et $Z_2 \leq Z_3$. Si $Z_1 = Z_2$ ou $Z_2 = Z_3$ alors on a évidemment $Z_1 \leq Z_3$. Si $Z_1 < Z_2$ et $Z_2 < Z_3$, alors on vérifie que $Z_1 < Z_3$.

□

Par exemple, dans le diagramme $\overline{\delta'}$,

$$\begin{aligned}
M \xleftarrow{bos} S \xrightarrow{mobos} G &\leq M \xleftarrow{b} B \xrightarrow{m_*} D,\\
M \xleftarrow{bos} S \xrightarrow{mobos} G &\leq D \xleftarrow{m_+} B \xrightarrow{mob} G,\\
M \xleftarrow{bos} S \xrightarrow{m_*os} D &\leq M \xleftarrow{b} B \xrightarrow{m_*} D,\\
D \xleftarrow{m_*os} S \xrightarrow{mobos} G &\leq D \xleftarrow{m_+} B \xrightarrow{mob} G.
\end{aligned}$$



$\overline{\delta'}$

Comme la relation $\leq$ est une relation d'ordre sur l'ensemble fini des zigzags élémentaires de $\overline{\delta}$, cet ensemble contient des éléments *maximaux*. Intuitivement, toute l'information sur les partages est contenue dans les zigzags élémentaires maximaux.

Définition 5.16 (Zigzag élémentaire maximal sur un diagramme complet $\overline{\delta}$)
Soit un diagramme complet $\overline{\delta}$. Un *zigzag élémentaire maximal* sur $\overline{\delta}$ est un élément maximal pour la relation d'ordre $\leq$ sur l'ensemble des zigzags élémentaires de $\overline{\delta}$.

Par exemple, les zigzags élémentaires

$$\begin{aligned}
M \xleftarrow{b} B \xrightarrow{m_*} D\\
D \xleftarrow{m_+} B \xrightarrow{mob} G
\end{aligned}$$

sont maximaux dans le diagramme $\overline{\delta'}$. Par contre, les zigzags élémentaires

$$\begin{aligned}
M \xleftarrow{bos} S \xrightarrow{mobos} G\\
M \xleftarrow{bos} S \xrightarrow{m_*os} D\\
D \xleftarrow{m_*os} S \xrightarrow{mobos} G
\end{aligned}$$

ne sont pas maximaux dans $\overline{\delta'}$.

Nous avons encore besoin de quelques définitions avant de donner un critère d'isomorphisme entre deux diagrammes. Nous commençons par définir un *nœud puits* dans un graphe. Un nœud puits est un nœud d'où ne part aucune flèche.

Définition 5.17 (Nœud puits) Soit un graphe δ^Φ. Un *nœud puits* de δ^Φ est un nœud $n \in \text{Nœud}(\delta^\Phi)$ qui vérifie la condition suivante : il n'existe aucun arc $a \in \text{Arc}(\delta^\Phi)$ tel que $\text{Source}(a) = n$.

Un *lien* sur un diagramme complet est un zigzag élémentaire maximal entre deux nœuds puits.

Définition 5.18 (Lien sur un diagramme complet) Soit un diagramme complet $\overline{\delta}$ sur $\mathcal{C}_0$. Un *lien* Z sur $\overline{\delta}$ est un zigzag élémentaire maximal sur $\overline{\delta}$

$$Z = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2$$

tel que m_0 et m_2 sont des nœuds puits de δ^Φ.

L'ensemble des liens d'un diagramme $\overline{\delta}$ est noté $\mathsf{Liens}(\overline{\delta})$.

Proposition 5.10 *L'ensemble des liens d'un diagramme complet $\overline{\delta}$ est calculable.*

Preuve. On calcule l'ensemble des zigzags élémentaires de $\overline{\delta}$. Cet ensemble est fini. Pour chaque zigzag élémentaire

$$Z_1 = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2,$$

on calcule l'ensemble des zigzags Z' tels que

- $\delta(a_0) \sim_{\overline{\delta}} \delta(a_1)$ $[Z']$
- Z' est formé d'une suite de zigzags élémentaires, c'est-à-dire est de la forme

$$Z' = n'_0 \xleftarrow{a'_0} n'_1 \xrightarrow{a'_1} n'_2 \ \ldots \ n'_{k'-2} \xleftarrow{a'_{k'-2}} n'_{k'-1} \xrightarrow{a'_{k'-1}} n'_{k'},$$

  où $\forall i \in \{0, \ldots, \frac{k'}{2} - 1\}$, $n_{2i} \xleftarrow{a'_{2i}} n'_{2i+1} \xrightarrow{a'_{2i+1}} n'_{2i+2}$ est un zigzag élémentaire.
- $\forall i \in \{0, \ldots, k'\}$, $m_1 \neq n'_i$.

Le zigzag élémentaire Z_1 est alors strictement plus petit que chaque zigzag élémentaire de Z'. Puis on ne conserve que les zigzags élémentaires maximaux entre nœuds puits. □

5.2.4 Forme minimale

Calculer la forme minimale d'un diagramme complet consiste à ne conserver que les nœuds puits, ainsi que les nœuds et les arcs qui font partie d'un lien du diagramme.

Définition 5.19 (Forme minimale) Soit un diagramme complet $\overline{\delta}$. La forme minimale du diagramme $\overline{\delta}$ est le diagramme

$$\overline{\beta} = \mathsf{Minimal}(\overline{\delta})$$

défini de la façon suivante.

- Graphe β^Φ :
  - $\text{Nœud}(\beta^\Phi) = \{n \in \text{Nœud}(\delta^\Phi)$, tel que n est un puits de δ^Φ ou n appartient à un lien de $\delta^\Phi\}$
  - $\text{Arc}(\beta^\Phi) = \{a \in \text{Arc}(\delta^\Phi)$, tel que a appartient à un lien de $\delta^\Phi\}$
- Foncteur $\beta : \beta^\Phi \to \mathcal{C}_0$.
  - $\forall n \in \text{Nœud}(\beta^\Phi)$, $\beta(n) = \delta(n)$;
  - $\forall a \in \text{Arc}(\beta^\Phi)$, $\beta(a) = \delta(a)$.

Proposition 5.11 *Pour tout diagramme complet $\overline{\delta}$,*

$$\overline{\delta} \cong \mathsf{Minimal}(\overline{\delta}).$$

Esquisse de la preuve. Soit $\overline{\beta} = \mathsf{Minimal}(\overline{\delta})$. On définit deux flèches de $\text{DIAGR}(\mathcal{C}_0)$, $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_\beta}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_\delta}$.

- On définit $\overline{\sigma} : \overline{\delta} \to \overline{\beta}$ de la façon suivante.
  - Morphisme généralisé de graphes $\sigma^\Phi : \delta^\Phi \rightsquigarrow \beta^\Phi$.
    - Pour tout nœud n de δ^Φ, tel que n est un nœud puits ou fait partie d'un lien de $\overline{\delta}$, on pose $\sigma^\Phi(n) = n$.
      Pour tout autre nœud n de δ^Φ, il existe un nœud puits n' et un arc $a_n : n \to n'$ dans δ^Φ. On pose $\sigma^\Phi(n) = n'$.
    - Pour tout arc $a : n \to n'$ de δ^Φ qui appartient à un lien de $\overline{\delta}$, on pose $\sigma^\Phi(a) = a$.
      Pour tout autre arc $a : n \to n'$ de δ^Φ, $\sigma^\Phi(a)$ est défini un peu plus loin.
  - Transformation naturelle généralisée $\sigma : \delta \rightsquigarrow \beta \circ \sigma^\Phi$.
    - Si n est un nœud de β^Φ, on pose $\sigma_n = id_{\delta(n)}$.
    - Sinon, on pose $\sigma_n = \delta(a_n)$ où $a_n : n \to n'$ est un arc de $\overline{\delta}$ tel que $n' = \sigma^\Phi(n)$.

Pour tout arc $a : n \to n'$ de δ^Φ, si a appartient à un lien de $\overline{\delta}$, on a alors évidemment

$$\sigma_{n'} \circ \delta(a) = \beta(a) \circ \sigma_n.$$

Si a n'appartient pas à un lien de $\overline{\delta}$, il existe un zigzag Z tel que

$$\sigma_{n'} \circ \delta(a) \sim_{\overline{\beta}} \sigma_n \ [Z].$$

On pose $\sigma^\Phi(a) = Z$.

- La flèche $\overline{\tau} : \overline{\beta} \to \overline{\delta}$ est définie de façon évidente.
- On a $\overline{\sigma} \circ \overline{\tau} = \overline{id_{\overline{\beta}}}$.
- On vérifie également que $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\overline{\delta}}}$.

□

Théorème 5.1 *Deux diagrammes complets sont isomorphes si et seulement si leurs formes minimales sont égales.*

Esquisse de la preuve. Si les formes minimales de deux diagrammes sont égales, alors les diagrammes sont évidemment isomorphes. Réciproquement, soit deux diagrammes $\overline{\alpha}$ et $\overline{\beta}$ sous forme minimale. Soit deux flèches $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ et $\overline{\tau} : \overline{\beta} \to \overline{\alpha}$ de DIAGR($\mathcal{C}_0$) telles que $\overline{\sigma} \circ \overline{\tau} \approx \overline{id_{\overline{\beta}}}$ et $\overline{\tau} \circ \overline{\sigma} \approx \overline{id_{\overline{\alpha}}}$.

1. On montre que pour tout nœud puits n de α^Φ, $\sigma^\Phi(n)$ est un nœud puits de β^Φ, $\alpha(n) = \beta(\sigma^\Phi(n))$ et $\sigma_n = id_{\alpha(n)}$.

2. Soit un zigzag élémentaire

$$m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2$$

de $\overline{\alpha}$. On pose $n_0 = \sigma^\Phi(m_0)$ et $n_2 = \sigma^\Phi(m_2)$. Les nœuds n_0 et n_2 sont donc des nœuds puits de β^Φ, et on a

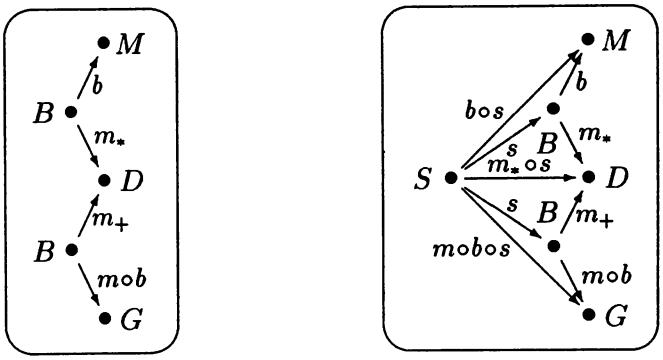
$$\alpha(m_0) = \beta(n_0),\ \alpha(m_2) = \beta(n_2),$$
$$\sigma_{m_0} = id_{\alpha(m_0)},\ \sigma_{m_2} = id_{\alpha(m_2)}.$$

On montre qu'il existe un nœud n_1 dans β^Φ tel que $\alpha(m_1) = \beta(n_1)$. Enfin, on montre qu'il existe deux arcs $b_0 : n_1 \to n_0$ et $b_1 : n_1 \to n_2$ dans β^Φ tel que $\alpha(a_0) = \beta(b_0)$ et $\alpha(a_1) = \beta(b_1)$.

3. Finalement, les diagrammes $\overline{\alpha}$ et $\overline{\beta}$ ont donc les mêmes nœuds puits, reliés par les mêmes liens. Les diagrammes $\overline{\alpha}$ et $\overline{\beta}$ sont donc identiques.

□

Exemple 5.12 Revenons aux diagrammes $\overline{\delta_2}$ et $\overline{\delta'}$.



$\overline{\delta_2}$ $\overline{\delta'}$

Le diagramme $\overline{\delta_2}$ est sous forme minimale ; le diagramme $\overline{\delta'}$ a pour forme minimale le diagramme $\overline{\delta_2}$.

$\mathsf{Minimal}(\overline{\delta_2}) = \overline{\delta_2}$;
$\mathsf{Minimal}(\overline{\delta'}) = \overline{\delta_2}$.

Finalement, les diagrammes $\overline{\delta_2}$ et $\overline{\delta'}$ sont donc isomorphes.

5.2.5 Algorithme

Finalement, un algorithme pour détecter si deux diagrammes sont isomorphes dans la catégorie diagr($\mathcal{C}_0$) consiste à calculer la complétion puis la forme minimale de chaque diagramme, et enfin à comparer les formes minimales. Pour calculer les formes minimales, il est nécessaire de calculer l'ensemble des liens associés à chaque diagramme.

Fonction Isomorphe ($\overline{\alpha_0}$, $\overline{\beta_0}$: diagramme) : booléen

$\overline{\alpha}$, $\overline{\beta}$: diagramme ;

Début

$\overline{\alpha}$:= Minimal(Complétion($\overline{\alpha_0}$)) ;
$\overline{\beta}$:= Minimal(Complétion($\overline{\beta_0}$)) ;
Résultat : $\overline{\alpha} = \overline{\beta}$;

Fin Isomorphe

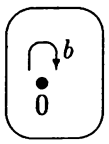
5.3 Cas général

Dans cette partie, nous examinons les problèmes soulevés lorsque la catégorie $\mathcal{C}_0$ est infinie ou contient des cycles.

5.3.1 Zigzags

Dans ce paragraphe, nous revenons un instant sur les zigzags. Considérons un diagramme fini $\overline{\delta}$. Le graphe δ^Φ est fini. Par contre, le nombre de zigzags sur δ^Φ est en général infini. En effet, rien n'interdit de passer plusieurs fois par le même arc.

- Si nous considérons un graphe avec une ou plusieurs boucles, il existe évidemment un nombre infini de zigzags sur ce graphe.



graphe ρ^Φ

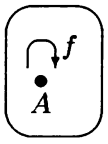
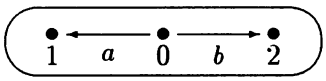


diagramme $\overline{\rho(A, f)}$

Nous pouvons en effet considérer les zigzags suivants.

$$0 \xrightarrow{b} 0$$
$$0 \xrightarrow{b} 0 \xrightarrow{b} 0$$
$$0 \xrightarrow{b} 0 \xrightarrow{b} 0 \cdots 0 \xrightarrow{b} 0$$

- Un diagramme, même s'il est construit sur un graphe sans cycle, contient en général une infinité de zigzags. Considérons par exemple le diagramme $\overline{\pi}$ construit sur le graphe π^Φ.



graphe π^Φ

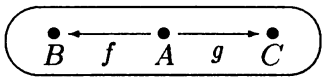


diagramme $\overline{\pi}$

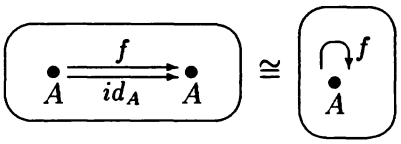
Nous pouvons considérer les zigzags suivants.

$$1 \xleftarrow{a} 0 \xrightarrow{b} 2$$
$$1 \xleftarrow{a} 0 \xrightarrow{b} 2 \xleftarrow{b} 0 \xrightarrow{a} 1$$
$$1 \xleftarrow{a} 0 \xrightarrow{b} 2 \xleftarrow{b} 0 \xrightarrow{a} 1 \xleftarrow{a} 0 \xrightarrow{b} 2$$
$$\cdots$$

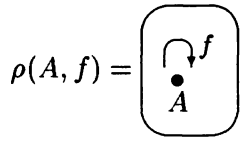
De façon générale, il n'existe pas forcément de borne supérieure pour la longueur des zigzags à considérer lorsqu'on veut tester si deux flèches u et v sont connectées, c'est-à-dire s'il existe un zigzag Z tel que $u \sim_{\overline{\delta}} v$ $[Z]$.

5.3.2 Catégorie de base comportant des cycles

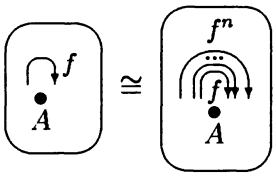
Nous supposons dans ce paragraphe que la catégorie de base $\mathcal{C}_0$ contient des cycles. Remarquons d'abord que la contraction d'un arc identité dans un diagramme peut former des *boucles* dans le graphe sous-jacent du diagramme, comme dans l'exemple 5.9–3.



Si $\mathcal{C}_0$ comporte des cycles, l'algorithme présenté en section 5.2 n'est pas correct. En effet, le principal de l'information contenue dans le diagramme n'est plus concentrée dans les nœuds puits. Certains diagrammes, comme $\rho(A, f)$ n'ont pas de nœud puits.



Si l'ensemble $\{f^n,\ n \in \mathbb{N}\}$ est infini, c'est-à-dire si toutes les flèches f^n sont différentes, alors on peut ajouter une infinité d'arcs au diagramme. On peut en effet ajouter un arc étiqueté par $f^2 = f \circ f$, un arc étiqueté par $f^3 = f \circ f \circ f$, etc. Autrement dit, pour tout entier n, les deux diagrammes ci-dessous sont isomorphes.



Par conséquent, si l'ensemble $\{f^n,\ n \in \mathbb{N}\}$ est infini, alors la fonction de complétion appliquée sur le diagramme $\rho(A, f)$ boucle.

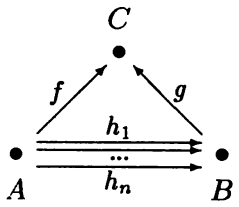
5.3.3 Catégorie de base infinie

Lorsque la catégorie de base $\mathcal{C}_0$ est infinie, en particulier lorsque $\mathcal{C}_0$ comporte un nombre infini de flèches entre deux objets, la procédure qui consiste à ajouter les factorisations à droite peut ne pas s'arrêter, ou, ce qui revient au même, conduire à un diagramme non fini.

Exemple 5.13 Soit deux flèches $f : A \to C$ et $g : B \to C$ de $\mathcal{C}_0$. Supposons qu'on a une famille indicée par $\mathbb{N}$ (donc infinie) de flèches

$$\{h_i : A \to B,\ i \in \mathbb{N}\}$$

telle que $\forall i \in \mathbb{N}$, $g \circ h_i = f$.



Si on considère un diagramme qui comporte deux arcs

$$n_0 \xrightarrow{a} n_1 \xleftarrow{b} n_2$$

étiquetés respectivement par f et g, on peut alors réaliser une infinité de factorisations à droite. Par conséquent, la fonction de complétion boucle.

Conclusion

1 Bilan

Nous avons proposé un cadre théorique pour étudier les spécifications algébriques modulaires. Nous avons repris une idée classique en spécification algébrique : la composition de spécifications peut être modélisée par des colimites de diagrammes finis. La construction principale est la somme amalgamée qui permet de représenter la composition de deux spécifications en précisant quelles parties sont partagées. Notre travail a consisté à poursuivre cette idée bien connue selon trois directions.

- D'un point de vue syntaxique, nous avons défini un langage de termes pour les constructions de colimites finies sur une catégorie de base $\mathcal{C}_0$. Ce langage est défini formellement par la catégorie Terme($\mathcal{C}_0$).
- D'un point de vue sémantique, nous avons proposé d'interpréter les termes par des diagrammes finis sur $\mathcal{C}_0$. Cette représentation est plus abstraite que la représentation par des termes car elle permet d'éliminer une partie des choix effectués lors de la construction de la spécification modulaire.
- Nous avons défini la notion d'*isomorphisme de construction* pour les spécifications modulaires. Un isomorphisme de construction entre deux spécifications modulaires correspond à un isomorphisme entre les diagrammes correspondants. Nous avons enfin proposé un algorithme pour détecter si deux diagrammes sont isomorphes, dans le cas particulier où la catégorie de base $\mathcal{C}_0$ est finie et ne comporte pas de cycle.

Notre travail est articulé en cinq chapitres.

1. Dans le chapitre 1, après quelques rappels sur les spécifications algébriques équationnelles, nous avons présenté plusieurs spécifications modulaires d'anneaux, qui nous ont permis d'introduire la définition d'isomorphisme de construction. Nous avons également précisé le cadre général de notre travail : la théorie des institutions, avec l'hypothèse que la catégorie des spécifications est finiment cocomplète.
2. Dans le chapitre 2, nous avons défini les notions de diagramme et morphisme de diagrammes en utilisant une formulation proche de l'informatique. Notre définition de diagramme est en effet basée sur celle de graphe, et non sur une catégorie quelconque. Nous avons reformulé les définitions de cône et de

colimite dans ce cadre. Nous avons défini deux catégories de diagrammes: DIAGR($\mathcal{C}_0$) et diagr($\mathcal{C}_0$). L'intérêt de la catégorie diagr($\mathcal{C}_0$) se situe sur le plan théorique puisque

- la catégorie diagr($\mathcal{C}_0$) est finiment cocomplète;
- la catégorie diagr($\mathcal{C}_0$) est une complétion de $\mathcal{C}_0$ par colimites finies.

La catégorie DIAGR($\mathcal{C}_0$), généralisation de certaines catégories utilisées en informatique — comme celle présentée dans [TBG91] —, est plus concrète que diagr($\mathcal{C}_0$) dans la mesure où elle permet de manipuler les morphismes de diagrammes de façon effective.

3. Le chapitre 3 est consacré à la définition de la précatégorie TERME($\mathcal{C}_0$), qui offre un langage pour les constructions modulaires. Les objets de TERME($\mathcal{C}_0$) sont des termes qui dénotent des spécifications modulaires, et les flèches de TERME($\mathcal{C}_0$) sont des termes qui dénotent des morphismes de spécifications modulaires. Nous proposons une construction stratifiée de cette précatégorie, afin de résoudre le problème de circularité entre la définition des flèches et de l'équivalence dans TERME($\mathcal{C}_0$). On obtient, en passant au quotient, une catégorie Terme($\mathcal{C}_0$). Nous avons montré que

   - la catégorie Terme($\mathcal{C}_0$) est finiment cocomplète;
   - la catégorie Terme($\mathcal{C}_0$) est une extension conservatrice de $\mathcal{C}_0$.

   Enfin, nous avons montré qu'on peut construire une catégorie $\mathcal{L}(\mathcal{C}_0)$ librement engendrée sur $\mathcal{C}_0$ par objet initial choisi et sommes amalgamées choisies. Cette catégorie est obtenue en quotientant TERME($\mathcal{C}_0$) par une congruence sur les objets et flèches de cette précatégorie.

4. Nous proposons dans le chapitre 4 une interprétation des termes représentant des spécifications modulaires par des diagrammes. Cette représentation, qui permet de faire abstraction de certaines étapes particulières choisies pour la construction de la spécification modulaire, est décrite par un préfoncteur

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

   qui correspond à un foncteur entre les catégories correspondantes

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

   Nous avons montré que

   - le foncteur $\mathcal{D}$ définit une équivalence entre les catégories Terme($\mathcal{C}_0$) et diagr($\mathcal{C}_0$);
   - deux objets de Terme($\mathcal{C}_0$) sont isomorphes si et seulement si leurs diagrammes associés sont isomorphes;
   - l'interprétation $\mathcal{D}$ est correcte, c'est-à-dire qu'une spécification modulaire est isomorphe à la colimite du diagramme qui la représente.

5. La représentation des constructions modulaires par des diagrammes, bien que plus abstraite que celle des termes ne permet pas de détecter immédiatement des isomorphismes de constructions. En effet, deux spécifications modulaires isomorphes sont représentées par des diagrammes qui sont isomorphes, mais pas forcément identiques. Dans le chapitre 5, nous proposons une normalisation des diagrammes, dans le cas particulier où la catégorie des spécifications de base est finie et ne comporte pas de cycle. Ayant montré que deux diagrammes sont isomorphes si et seulement si ils ont la même forme normale, nous disposons donc d'une procédure pour décider si deux diagrammes sont isomorphes.

Le formalisme proposé est *compositionnel*, dans la mesure où des compositions successives de spécifications peuvent être vues comme un assemblage unique. En effet, une suite de constructions de sommes amalgamées est équivalente à la colimite d'un diagramme plus complexe. Cette propriété provient de la structure de *monade* des précatégories TERME($\mathcal{C}_0$) et DIAGR($\mathcal{C}_0$).

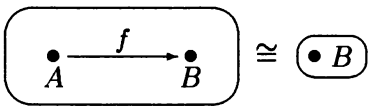
Nous espérons avoir montré que la modularité, en particulier le problème des partages entre plusieurs spécifications, est correctement modélisé par les colimites finies. En effet, l'utilisation des colimites permet de faire abstraction d'une part de la définition effective des spécifications et d'autre part des étapes particulières choisies pour la construction d'une spécification modulaire. On peut ainsi se concentrer uniquement sur l'assemblage des différentes spécifications de base.

2 Perspectives

Citons quelques voies possibles pour poursuivre ce travail.

Contraintes sémantiques

De nombreux langages de spécification permettent de préciser des *contraintes sémantiques*. Un exemple de contrainte sémantique est la *persistance* d'un foncteur de synthèse entre deux classes d'algèbres. Cette contrainte assure la préservation par exemple d'un module importé, ou encore du paramètre effectif lors d'une instanciation (cf. par exemple [EM85]). Une spécification B qui importe une spécification A est une flèche $f : A \to B$. J.-C. Reynaud [Rey90b, Rey93] a proposé de *distinguer* les flèches auxquelles est associée une contrainte sémantique. Dans notre cadre, l'utilisation de morphismes distingués ne paraît pas suffisante. En effet, comme les deux diagrammes suivants sont isomorphes, l'information d'importation entre A et B est perdue lors du passage à la colimite.



Par conséquent, le concept de colimite, qui convient à la modélisation des partages, ne peut pas être appliqué directement pour prendre en considération des contraintes sur des morphismes de spécifications.

Enrichissement et abstraction

Deux opérateurs de constructions de spécifications sont fondamentaux en spécification algébrique : l'*enrichissement*, qui permet à partir d'une spécification d'obtenir une nouvelle spécification en ajoutant des sortes et des opérateurs ; et l'*abstraction*, qui permet d'obtenir une nouvelle spécification en supprimant des sortes et des opérateurs.

Il semble possible d'étendre le formalisme proposé afin de prendre en compte les enrichissements. Pour nous, un enrichissement d'une spécification modulaire S est une spécification S' accompagnée d'un morphisme de spécifications $p : S \to S'$ qui modélise l'inclusion de S dans S'. La spécification S fait partie de Terme($\mathcal{C}_0$), mais pas S', ni par conséquent f. Une solution pour permettre d'ajouter de telles flèches est de considérer la catégorie T' qui contient Terme($\mathcal{C}_0$), l'objet S et la flèche f. Ensuite, on considère la sous-catégorie pleine $\mathcal{C}'_0$ de T' qui a pour objets d'une part l'objet S et d'autre part les objets de $\mathcal{C}_0$. Finalement, la catégorie Terme($\mathcal{C}'_0$) contient Terme($\mathcal{C}_0$), ainsi que toutes les constructions qui ont été ajoutées par l'introduction de l'enrichissement S'. La flèche $p : S \to S'$ appartient bien à la catégorie Terme($\mathcal{C}'_0$) car celle-ci a pu être reconstruite à l'aide des flèches **up**.

Le problème de l'abstraction semble plus difficile à résoudre. Dans notre cadre, une abstraction d'une spécification S de Terme($\mathcal{C}_0$) est une spécification S' accompagnée d'un morphisme de spécifications $q : S' \to S$. Dans ce cas, la construction décrite dans le cas de l'enrichissement ne fonctionne pas, car la flèche q n'appartient pas à la nouvelle catégorie des termes Terme($\mathcal{C}'_0$). Une voie de recherche pourrait être ici la prise en considération de colimites distinguées dans la catégorie de base $\mathcal{C}_0$.

Colimites distinguées dans $\mathcal{C}_0$

En théorie des esquisses [Ehr68, DR94a, DR94b], une esquisse contient des cônes et des cocônes *distingués*. Le type d'une esquisse correspond à une catégorie librement engendrée par sommes et produits, en conservant les limites et colimites spécifiées.

Dans l'état actuel de notre travail, s'il existe par exemple une somme amalgamée dans la catégorie de base $\mathcal{C}_0$, cette somme est "perdue" lors de la construction de la catégorie Terme($\mathcal{C}_0$). Il serait intéressant de généraliser la construction de Terme($\mathcal{C}_0$) de façon à pouvoir conserver certaines colimites.

La prise en considération de certaines colimites dans la catégorie de base peut résoudre le problème de l'abstraction. En effet, on peut imaginer ajouter à la catégorie de base la spécification S ainsi que les colimites qui ont permis de construire S, puis d'ajouter S' et le morphisme de spécifications $q : S' \to S$. On obtient ainsi une catégorie $\mathcal{C}'_0$, qui contient comme colimites distinguées d'une part les colimites distinguées de $\mathcal{C}_0$, et d'autre part les colimites qui ont permis de construire S. Finalement, la catégorie Terme($\mathcal{C}'_0$) contient Terme($\mathcal{C}_0$), S', $q : S' \to S$, ainsi que toutes les constructions qui utilisent S'.

Généralisation de l'algorithme

En ce qui concerne les diagrammes, nous avons proposé un critère pour décider si deux diagrammes sont isomorphes dans diagr($\mathcal{C}_0$). Il resterait à évaluer la complexité de cet algorithme. D'autre part, nous avons supposé que la catégorie des spécifications de base $\mathcal{C}_0$ est finie et ne comporte pas de cycle. Cette hypothèse est compatible avec le langage de spécification LPG. Il serait néanmoins intéressant soit de généraliser l'algorithme proposé, soit de montrer l'indécidabilité de ce problème.

Application aux langages de spécification et de programmation

Notre travail a des retombées pratiques sur le traitement de la modularité dans les langages de spécification et de programmation.

Tout d'abord, la somme amalgamée est couramment utilisée dans les langages de spécification algébrique ; par contre les flèches "**up**", à part dans le langage LPG, sont ignorées. D'un point de vue théorique, cela implique que les catégories considérées ne sont pas finiment cocomplètes. D'un point de vue pratique, cela signifie qu'on ne peut pas modéliser toutes les compositions de spécifications, en particulier certains assemblages décrits dans le chapitre 1 sur l'exemple des anneaux.

D'autre part, le langage de termes proposé, qui permet de coder finement les assemblages de modules, a l'avantage d'introduire une gestion explicite des partages entre spécifications. Remarquons également que l'instanciation des modules génériques peut être traitée de façon uniforme dans le même cadre.

Ces éléments doivent permettre de définir, pour les langages de la prochaine génération, un concept de modularité plus général et bien fondé d'un point de vue théorique.

Réutilisation

Enfin, le formalisme que nous avons proposé peut également être appliqué à la réutilisation de programmes. On suppose que l'on dispose d'une bibliothèque de spécifications de base, accompagnées de différentes implantations. Comme il peut être nécessaire d'implanter une même spécification de façons différentes dans diverses parties du programme, par exemple pour des raisons d'efficacité, cette bibliothèque doit contenir différentes implantations pour chaque spécification de base. D'autre part, cette bibliothèque peut également contenir des implantations de spécifications modulaires construites sur les spécifications de base.

L'objectif est d'implanter une nouvelle spécification modulaire, en réutilisant au mieux les implantations disponibles. Si cette spécification est isomorphe à une construction déjà implantée, on peut bien évidemment réutiliser cette implantation. De façon plus générale, il s'agit de détecter les implantations réutilisables et les spécifications qui restent à implanter, et, parallèlement, les contraintes posées sur ces implantations par les partages entre les spécifications de base utilisées. L'idée est d'extraire des informations du diagramme associé à la spécification modulaire à implanter, ce diagramme pouvant être considéré comme le degré de liberté dont on dispose pour coder les spécifications de base.

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 5^{th} 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.

[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.

[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 troisième 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