LSR

Laboratoire Logiciels, Systèmes, Réseaux

RAPPORT DE RECHERCHE

Modular specifications: constructions with finite colimits, diagrams, isomorphisms

Catherine Oriat

RR 964-I-LSR 3 **Novembre 1996**

BP 53 - 38041 Grenoble Cedex 9 - France

Fédération IMAG
Centre National de la Recherche Scientifique - Institut National Polytechnique de Grenoble -
Université Joseph Fourier Grenoble I

Modular specifications: constructions with finite colimits, diagrams, isomorphisms

Catherine ORIAT

Résumé : La composition de spécifications modulaires peut être modélisée, dans le formalisme des catégories, par des colimites de diagrammes. Les sommes amalgamées permettent en particulier de décrire l'assemblage de deux spécifications qui ont une partie commune. Ce travail étend cette idée classique selon trois axes.

Tout d'abord, nous définissons un langage de termes pour représenter les spécifications modulaires construites à l'aide de colimites sur une catégorie de base. Formellement, ce langage est caractérisé par une catégorie finiment cocomplète.

Nous proposons ensuite d'associer à chaque terme un diagramme. Cette interprétation permet de faire abstraction de certains choix effectués lors de la construction de la spécification modulaire. Nous définissons une catégorie de diagrammes, qui est une complétion de la catégorie de base par colimites finies. Nous montrons que cette interprétation définit une équivalence entre la catégorie des termes et la catégorie des diagrammes, ce qui montre la correction de l'interprétation.

Enfin, nous proposons un algorithme pour normaliser les diagrammes, dans le cas où la catégorie de base est squelettique, finie et sans cycle. Cela nous permet de détecter des "isomorphismes de construction" entre spécifications modulaires, c'est-à-dire des isomorphismes qui ne dépendent pas de la sémantique des spécifications de base, mais seulement de leur assemblage.

Abstract: The composition of modular specifications can be modeled, in a category theoretic framework, by colimits of diagrams. Pushouts in particular describe the combination of two specifications sharing a common part. This work extends this classic idea along three lines.

First, we define a term language to represent modular specifications built with colimit constructions over a category of base specifications. This language is formally characterized by a finitely cocomplete category.

Then, we propose to associate with each term a diagram. This interpretation provides us with a more abstract representation of modular specifications because irrelevant steps of the construction are eliminated. We define a category of diagrams, which is a completion of the base category with finite colimits. We prove that the interpretation of terms as diagrams defines an equivalence between the corresponding categories, which shows the correctness of this interpretation.

At last, we propose an algorithm to normalize diagrams, in the case when the base category is skeletal, finite and cycle free. This allows us to detect "construction isomorphisms" between modular specifications, i.e. isomorphisms which do not depend on the semantics of the base specifications, but only on their combination.

Modular specifications: constructions with finite colimits, diagrams, isomorphisms

Catherine ORIAT
LSR–IMAG
BP 53
38041 Grenoble Cedex 09, France
e-mail : Catherine.Oriat@imag.fr

1 Introduction

The specification of large programs requires to split up the problem to solve into several simpler problems. This top-down approach, corresponding to the slogan *divide and conquer*, is classic in software engineering. The decomposition of specifications into *modules* allows one to reflect the logical structure of the problem, and to develop the different parts of the program independently.

On the opposite, we are interested here in the *composition* of specifications. This bottom-up approach to modularity consists in storing elementary specifications in a library and in constructing new specifications by gathering already defined ones. Such a reuse of specifications (and possibly of programs as well) is advocated in software engineering to avoid errors and reduce development costs.

We focus on algebraic specifications [16, 26, 37], which come from universal algebra in mathematics, and from abstract data types in software engineering. Our aim is to study the composition of algebraic specifications in a category theoretic framework. Indeed, the algebraic specification formalism strongly relies on category theory. Historically, the development of algebraic specification, in particular the work of the ADJ group [22, 23, 21], has been influenced by Lawvere's algebraic theories [25, 6, 36]. Practically, the use of category theory may be justified by the important role played by *specification morphisms* for structuring specifications. Moreover, the combination of specifications related by specification morphisms can be modeled using the categorical concept of *colimit*. Intuitively, a *diagram* describes a combination of objects with some sharings, and the *colimit* of a diagram the result of this combination.

We suppose that we have a library of elementary specifications and specification morphisms, which forms a *base category* $\mathcal{C}_0$. We define a *term language* to express specifications and specification morphisms built from the base category with colimit constructions. Then, we propose to associate with each term denoting a specification a *diagram*, and with each term denoting a specification morphism a *diagram morphism*. We thus need to define a *category* of diagrams. This representation of modular specifications is more abstract than terms in that it allows us to get rid of some specific steps chosen for the construction of the modular specification. At last, we propose a procedure to decide whether two diagrams are isomorphic, in the case when the category $\mathcal{C}_0$ of base specifications is skeletal, finite and cycle free. This allows us to detect *construction isomorphisms* between specifications, by testing whether their associated diagrams are isomorphic. These isomorphisms do not depend on the contents of the base specifications, but only on their combinations.

Algebraic specifications can be based on various logics, for instance equational, Horn-clause or first-order logic. Goguen and Burstall have developed the *theory of institutions* [19, 20] to formalize a notion of *logical system* which is independent of the underlying logic. As the problem of composing specifications does not depend on the logic used to express specifications, we can work in the framework of institutions. We only need to assume that the category of specifications is finitely cocomplete, which means that every finite diagram has a colimit. Goguen and Burstall have shown [20] that the category of specifications is finitely cocomplete if and only if the category of *signatures* is finitely cocomplete.

The idea of modeling the interconnection of systems by means of colimits was proposed by Goguen [17, 18]. Then, colimits were used in the context of algebraic specification to describe the semantics of the specification language Clear [7, 8]. The diagram morphisms used in this context were mere *inclusions of diagrams*. This idea has been extended to more general categories of diagrams such as for instance the flatten category $\mathbf{Func}(\mathcal{C}_\mathbf{0})$ described in [34], page 244. However, these categories of diagrams are not general enough to model *any* combination of specifications because they are not, in the general case, *finitely cocomplete*. We propose here to work in a category $\mathrm{Diagr}(\mathcal{C}_0)$ of diagrams which is finitely cocomplete. Similarly, from a syntactic point of view, term languages which have been proposed until now to express colimit constructions are not based on a finitely cocomplete category either (see for instance [15]). The term language proposed here relies on a finitely cocomplete category $\mathrm{Term}(\mathcal{C}_0)$, and therefore is powerful enough to express any combination of specifications.

Syntax and representation choices

In computer science, we usually distinguish between the *syntax*, which corresponds to the *language* used to describe the entities (e.g. a program), and the *semantics*, which corresponds to the *meaning* of the entities (e.g. the result of a program execution). The syntax requires representation choices, while the semantics may give a result which is independent of the coding. In category theory, no representation choices are made on arrows of a category in the sense that two syntactically different arrows may be equal. For example, in a category we have $f \circ id_A = f$ for every arrow $f : A \to B$. From a syntactic point of view, this is not satisfactory because we want to make a distinction between a syntactic equality (i.e. identity of symbol strings: $f \circ id_A = f \circ id_A$) and a semantic equality (i.e. equivalence of meaning: $f \circ id_A = f$). We thus use *equiv-categories*.

An equiv-category has the same structure as a category, except that equalities between arrows are replaced by a congruence relation, introduced to model semantic equivalence. Considering the quotient of an equiv-category by its associated congruence yields a category which might be considered as a (trivial) semantics. So, in an equiv-category, we have $f \circ id_A = f \circ id_A$ and $f \circ id_A \sim f$. Equiv-categories were introduced by D. Duval and J.-C. Reynaud [11]. Usual concepts of category theory can be restated in the context of equiv-categories: *equiv-functors* correspond to functors, *equiv-colimits* to colimits ... etc[1].

Problems of representation choices may also occur at the level of objects when categorical concepts are defined *up to isomorphism*. In particular, colimits are usually defined up to isomorphism: a diagram may have several colimits which are then isomorphic. The set $(\{1\} \times A) \cup (\{2\} \times B)$ is one example of sum of A and B in the category of sets. If we want to define a syntax to represent colimit constructions, we can no longer define colimits up to isomorphism, because we must make representation choices for these constructions. We must consider *chosen colimits*, i.e. canonical colimits among all isomorphic colimits. We might for instance decide that $(\{1\} \times A) \cup (\{2\} \times B)$ is *the chosen sum* of A and B in the category of sets. Theoretically, these choices are needed if we want to get a *free* construction of terms. The introduction of *chosen colimits*, due to C. Ehresmann [13, 14], endows a category with

[1] In [28], equiv-categories, equiv-functors and equiv-colimits were respectively called *precategories*, *prefunctors* and *precolimits*. We now adopt D. Duval and J.-C. Reynaud's terminology.

an algebraic structure, the chosen colimits playing the role of *constructors.* Of course, in an equiv-category we actually need to consider *chosen equiv-colimits.*

This paper is organized as follows. In section 2, we present various modular specifications of rings constructed from a category $\mathcal{C}_0$ of base specifications. This academic example presents on the one hand a syntax for modular specifications and on the other hand the notion of construction isomorphism between two modular specifications.

Section 3 contains the theoretical bases of our work. We present the notion of equiv-category as well as the equiv-category of diagrams $\mathrm{DIAGR}(\mathcal{C}_0)$. We show that $\mathrm{DIAGR}(\mathcal{C}_0)$ is finitely cocomplete, and is a completion of $\mathcal{C}_0$ with finite equiv-colimits.

Section 4 is dedicated to the formal definition of the syntax for modular specifications. We present a stratified construction of an equiv-category $\mathrm{TERM}(\mathcal{C}_0)$ which provides us with a term language to represent modular specifications and specification morphisms. We show that $\mathrm{TERM}(\mathcal{C}_0)$ is finitely cocomplete, is a conservative extension of $\mathcal{C}_0$, and is freely generated over $\mathcal{C}_0$ by a chosen equiv-initial object and chosen equiv-pushouts.

In section 5, we associate terms with diagrams and show that this interpretation defines an equivalence between the equiv-categories $\mathrm{DIAGR}(\mathcal{C}_0)$ and $\mathrm{TERM}(\mathcal{C}_0)$.

In section 6, we suppose that the base category $\mathcal{C}_0$ is skeletal, finite and cycle free. In that case, we define a normalization procedure for diagrams. This procedure allows us to detect construction isomorphisms between two modular specifications by comparing the normal forms of their corresponding diagrams.

2 Example

In this section, we present various ways of specifying the theory of rings with modular specifications. This example only makes use of *equational* specifications, but we could actually work in any institution whose category of specifications is finitely cocomplete.

2.1 Base specifications

First of all, we define a library of *base specifications.* Base specifications are written in the specification language LPG [3, 4]. The LPG language has two kinds of specifications: *types* which are interpreted as *initial algebras* (or, for generic types, as *free algebras*), and *properties* which are interpreted as *classes of algebras satisfying the equations.* In the example developed here, we will only use properties, which means that a model of a specification is the class of all algebras which satisfy the equations. We can also define specification morphisms with a statement introduced by the keyword `satisfies`. The semantics of LPG is described in [29], and the rules to compose specification morphisms are presented in [5].

We now specify the base specifications and base specification morphisms.

$S =$

```
property   A-SORT
sorts      s
```

$B =$

```
property    BIN-OP
sorts       s
operators   op :  s,s -> s
satisfies   A-SORT[s]
```

The property S specifies a single sort `s`. The property B specifies a single sort, also called `s`, a binary operator `op` and a specification morphism $s : S \to B$, defined by the statement `satisfies A-SORT[s]` in the module B. This morphism maps the sort `s` of S to the sort `s` of B.

$M =$

```
property   MONOID
sorts      s
operators  * :  s,s -> s
           1 :  -> s
equations  1 * x == x
           x * 1 == x
 (x * y) * z == x * (y * z)
satisfies  BIN-OP[s,*]
```

$G =$

```
property   ABEL-GROUP
sorts      s
operators  + :  s,s -> s
           0 :  -> s
           i :  s -> s
equations  x + y == y + x
           i(x) + x == 0
satisfies  MONOID[s,+,0]
```

The property M specifies a monoid, with a specification morphism $b : B \to M$, which maps the sort `s` of B to the sort `s` of M and the operator `op` of B to the operator `*` of M. The property G specifies an Abelian group: we add to the specification of monoids an inverse function `i` and the commutativity of the binary operator. We also define a specification morphism $m : M \to G$. This morphism maps the operator `*` of M to the operator `+` of G and the constant `1` of M to the constant `0` of G.

$D =$

```
property   DISTRIBUTIVE
sorts      s
operators  + :  s,s -> s
           * :  s,s -> s
equations  x * (y + z) == (x * y) + (x * z)
satisfies  BIN-OP[s,+], BIN-OP[s,*]
```

At last, the property D specifies two binary operators related by the distributive law, as well as two specification morphisms $m_+, m_* : B \to D$. The morphism m_+ maps `op` to `+` and the morphism m_* maps `op` to `*`.

We have so far defined a graph and now consider the category $\mathcal{C}_0$ freely generated by this graph. The category $\mathcal{C}_0$, called the *base category*, represents a *library* of available specifications and specification morphisms.

$$\begin{tikzcd}
S \arrow[r, "s"] & B \arrow[r, "b"] \arrow[d, shift right, "m_+"'] \arrow[d, shift left, "m_*"] & M \arrow[r, "m"] & G \\
 & D & &
\end{tikzcd}$$

Graph corresponding to the base category $\mathcal{C}_0$

2.2 *Combinations with colimit constructions*

We can combine base specifications to form new *modular* specifications. In LPG, we can for example write a specification R_1 of *rings* by combining the properties `MONOID`, `DISTRIBUTIVE` and `ABEL-GROUP` as follows.

$R_1 =$

```
property   RING
sorts      s
operators  +, * :  s, s -> s
           0, 1 :  -> s
           i :  s -> s
combines   MONOID[s,*,1],
           DISTRIBUTIVE[s,+,*],
           ABEL-GROUP[s,+,0,i]
```

This specification implicitly describes some sharings of sorts and operators. For example, the three specifications M, D and G share the same sort s. These sharings are of course crucial to get a specification of rings.

In our work, the sharings are expressed by *colimits of diagrams*. This categorical construction allows us to model the composition of objects with explicit sharings. In the example of rings, the specification R_1 corresponds to the colimit of the diagram $\overline{\delta_1}$.

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[dr, "m_*" near start] & M \\
S \arrow[ur, "b \circ s" near start] \arrow[r, "m_* \circ s"] \arrow[dr, "m \circ b \circ s"' near start] & D \\
B \arrow[ur, "m_+"' near start] \arrow[r, "m \circ b"'] & G
\end{tikzcd}$$

$\overline{\delta_1}$

Here is the intuitive interpretation of this diagram.

- The vertices labeled by M, D and G, which are *terminating vertices* of the graph (no edge has one of these vertices as source), are the specifications we wish to gather.

- The vertices labeled by B, S and B express the sharings.

  - The sort s is shared by the three specifications M, D and G. This sharing is modeled by the morphisms $b \circ s : S \to M$, $m_* \circ s : S \to D$ and $m \circ b \circ s : S \to G$.
  - The binary operator * is shared by the specifications M and D. This sharing is modeled by the morphisms $b : B \to M$ and $m_* : B \to D$.
  - The binary operator + is shared by the specifications G and D. This sharing is modeled by the morphisms $m \circ b : B \to G$ and $m_+ : B \to D$.

In the following, we focus on one particular construction of colimit: *pushouts*, which model the composition of two specifications.

Definition 1 (Pushout) Let A, B, C be three specifications and $f : A \to B$, $g : A \to C$ be two specification morphisms. A *pushout* of f and g is a triple

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

where $\mathsf{push}(A,B,C,f,g)$ is a specification and

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

are two specification morphisms, such that

1. $\mathsf{k}_1(A,B,C,f,g) \circ f = \mathsf{k}_2(A,B,C,f,g) \circ g$;

2. for any specification D and specification morphisms $f' : B \to D$, $g' : C \to D$ such that $f' \circ f = g' \circ g$, there exists a unique specification morphism

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

   such that
   $$\begin{cases} \mathsf{up}(A,B,C,D,f,f',g,g') \circ \mathsf{k}_1(A,B,C,f,g) = f' \\ \mathsf{up}(A,B,C,D,f,f',g,g') \circ \mathsf{k}_2(A,B,C,f,g) = g'. \end{cases}$$

In the following, if $P = \mathsf{push}(A,B,C,f,g)$, we write $\mathsf{k}_i(P)$ for $\mathsf{k}_i(A,B,C,f,g)$.

$$\begin{tikzcd}
 & B \arrow[dr, "\mathsf{k}_1(P)"'] \arrow[drr, "f'"] & & \\
A \arrow[ur, "f"] \arrow[dr, "g"'] & & P \arrow[r, "\mathsf{up}"] & D \\
 & C \arrow[ur, "\mathsf{k}_2(P)"] \arrow[urr, "g'"'] & &
\end{tikzcd}$$

A well known result of category theory states that a category is finitely cocomplete (i.e. every finite diagram has a colimit) if and only if this category has an initial object and pushouts.

This means that together with the empty specification, pushouts can simulate the colimit construction of any finite diagram. Indeed, the empty specification, noted $\emptyset$, is initial because for any specification A, there is a unique specification morphism from $\emptyset$ to A, noted $\mathsf{j}(A) : \emptyset \to A$.

We now may define a specification R_2 of rings with pushouts by combining the specifications M and D, and then adding G to the result:

$$\begin{aligned} &MD = \mathsf{push}(B, M, D, b, m_*) \\ &R_2 = \mathsf{push}(B, MD, G, \mathsf{k}_2(MD) \circ m_+, m \circ b) \end{aligned}$$

There are other ways of specifying a ring with the given base specifications. For instance, we can first combine D and G and then add the specification M. We get a new specification R_2' of rings.

$$\begin{aligned} &DG = \mathsf{push}(B, D, G, m_+, m \circ b) \\ &R_2' = \mathsf{push}(B, M, DG, b, \mathsf{k}_1(DG) \circ m_*) \end{aligned}$$

$$\begin{tikzcd}
 & M \arrow[dr, "k_1(MD)"] & & \\
B \arrow[ur, "b"] \arrow[dr, "m_*"'] & & MD \arrow[dr, "k_1(R_2)"] & \\
 & D \arrow[ur, "k_2(MD)"'] & & R_2 \\
B \arrow[ur, "m_+"] \arrow[dr, "m \circ b"'] & & & \\
 & G \arrow[uurr, "k_2(R_2)"'] & &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & M \arrow[ddrr, "k_1(R'_2)"] & & \\
B \arrow[ur, "b"] \arrow[dr, "m_*"'] & & & \\
 & D \arrow[dr, "k_1(DG)"] & & R'_2 \\
B \arrow[ur, "m_+"] \arrow[dr, "m \circ b"'] & & DG \arrow[ur, "k_2(R'_2)"'] & \\
 & G \arrow[ur, "k_2(DG)"'] & &
\end{tikzcd}$$

One can easily convince oneself that the specifications R_2 and R'_2 are isomorphic. Intuitively, both constructions are equivalent because the pushout operation is "associative" (in a sense which should be defined formally). Actually, R_2 and R'_2 are two different encodings of the colimit of the diagram $\overline{\delta_2}$ with pushouts.

$$\begin{tikzcd}
 & \bullet M \\
B \bullet \arrow[ur, "b"'] \arrow[dr, "m_*"] & \\
 & \bullet D \\
B \bullet \arrow[ur, "m_+"'] \arrow[dr, "m \circ b"] & \\
 & \bullet G
\end{tikzcd}$$

$\overline{\delta_2}$

But more complicated cases may arise. We can for instance start by defining a "pseudo-ring" i.e. a ring without distributivity either with the term P or P' as follows.

$$P = \mathit{push}(S, M, G, b \circ s, m \circ b \circ s)$$

$$Q_1 = \mathit{push}(S, B, B, s, s)$$
$$Q_2 = \mathit{push}(B, M, Q_1, b, \mathsf{k}_1(Q_1))$$
$$P' = \mathit{push}(B, Q_2, G, \mathsf{k}_2(Q_2) \circ \mathsf{k}_2(Q_1), m \circ b)$$

$$\begin{tikzcd}
 & M \arrow[dr, "k_1(P)"] & \\
S \arrow[ur, "b \circ s"] \arrow[dr, "m \circ b \circ s"'] & & P \\
 & G \arrow[ur, "k_2(P)"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & M \arrow[dr, "k_1(Q_2)"] & & \\
 & B \arrow[ur, "b"] \arrow[dr, "k_1(Q_1)"] & & Q_2 \arrow[dr, "k_1(P')"] & \\
S \arrow[ur, "s"] \arrow[dr, "s"'] & & Q_1 \arrow[ur, "k_2(Q_2)"'] & & P' \\
 & B \arrow[ur, "k_2(Q_1)"'] \arrow[dr, "m \circ b"'] & & & \\
 & & G \arrow[uurr, "k_2(P')"'] & &
\end{tikzcd}$$

The specifications P and P' correspond to the colimit of the diagram $\overline{\zeta}$ below. Now we can "add the distributivity" on two different ways and get two new specifications of rings R_3 and R_4. We first consider the specification

$$B_2 = \mathit{push}(\emptyset, B, B, j(B), j(B))$$

which groups two binary operators, and the specification morphism

$$u_1 = \mathit{up}(\emptyset, B, B, P, j(B), j(B), k_1(P) \circ b, k_2(P) \circ m \circ b) : B_2 \to P.$$

This arrow exists because as $\emptyset$ is initial, $k_1(P) \circ b \circ j(B) = k_2(P) \circ m \circ b \circ j(B)$. There is also an arrow $u_2 = \mathit{up}(\emptyset, B, B, D, j(B), j(B), m_*, m_+) : B_2 \to D$.

$$\begin{tikzcd}
 & & M \arrow[ddr, "k_1(P)"] & \\
 & B \arrow[ur, "b"] \arrow[dr, "k_1(B_2)"] & & \\
\emptyset \arrow[ur, "j(B)"] \arrow[dr, "j(B)"'] & & B_2 \arrow[r, "u_1"] & P \\
 & B \arrow[ur, "k_2(B_2)"'] \arrow[dr, "m \circ b"'] & & \\
 & & G \arrow[uur, "k_2(P)"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & B \arrow[dr, "k_1(B_2)"'] \arrow[drr, "m_*"] & & \\
\emptyset \arrow[ur, "j(B)"] \arrow[dr, "j(B)"'] & & B_2 \arrow[r, "u_2"] & D \\
 & B \arrow[ur, "k_2(B_2)"] \arrow[urr, "m_+"'] & &
\end{tikzcd}$$

We then obtain a specification R_3 of rings by combining D and P sharing B_2.

$$\begin{aligned}
R_3 &= \mathit{push}(B_2, P, D, u_1, u_2) \\
&= \mathit{push}(\mathit{push}(\emptyset, B, B, j(B), j(B)), P, D, \\
&\qquad\qquad \mathit{up}(\emptyset, B, B, P, j(B), j(B), k_1(P) \circ b, k_2(P) \circ m \circ b), \\
&\qquad\qquad \mathit{up}(\emptyset, B, B, D, j(B), j(B), m_*, m_+))
\end{aligned}$$

And here is a last specification R_4 of rings which uses P':

$$\begin{aligned} R_4 = \mathsf{push}(&\mathsf{push}(\emptyset, B, B, \mathsf{j}(B), \mathsf{j}(B)), P', D, \\ &\mathsf{up}(\emptyset, B, B, P', \mathsf{j}(B), \mathsf{j}(B), \mathsf{k}_1(P') \circ \mathsf{k}_2(Q_2) \circ \mathsf{k}_1(Q_1), \\ &\quad \mathsf{k}_1(P') \circ \mathsf{k}_2(Q_2) \circ \mathsf{k}_2(Q_1)), \\ &\mathsf{up}(\emptyset, B, B, D, \mathsf{j}(B), \mathsf{j}(B), m_*, m_+)) \end{aligned}$$

The constructions R_3 and R_4 respectively correspond to the following diagrams $\overline{\delta_3}$ and $\overline{\delta_4}$.

$$\begin{tikzcd} & M \\ S \arrow[ur, "b \circ s"] \arrow[dr, "m \circ b \circ s"'] & \\ & G \end{tikzcd}$$

$\overline{\zeta}$

$$\begin{tikzcd} & & M \\ & B \arrow[ur, "b"'] \arrow[dr, "m_*"] & \\ S \arrow[uurr, "b \circ s"] \arrow[ddrr, "m \circ b \circ s"'] & & D \\ & B \arrow[ur, "m_+"'] \arrow[dr, "m \circ b"] & \\ & & G \end{tikzcd}$$

$\overline{\delta_3}$

$$\begin{tikzcd} & & M \\ & B \arrow[ur, "b"'] \arrow[dr, "m_*"] & \\ S \arrow[ur, "s"'] \arrow[dr, "s"] & & D \\ & B \arrow[ur, "m_+"'] \arrow[dr, "m \circ b"] & \\ & & G \end{tikzcd}$$

$\overline{\delta_4}$

By using the definition of colimit in category theory, we can check that the colimits of the diagrams $\overline{\delta_1}$, $\overline{\delta_2}$, $\overline{\delta_3}$ and $\overline{\delta_4}$ are isomorphic, because of the equality of specification morphisms

$$m_+ \circ s = m_* \circ s.$$

This equality means that the fact that both operators + and * operate on the same set is contained in the specification D of distributive operators. We will see in section 3 and 6 that $\overline{\delta_1}$, $\overline{\delta_2}$, $\overline{\delta_3}$ and $\overline{\delta_4}$ are isomorphic in the category of diagrams $\mathrm{Diagr}(\mathcal{C}_0)$.

2.3 Construction isomorphism

We have just seen that there are various equivalent ways of specifying the theory of rings from a given category of base specifications. Formally, two specifications are equivalent if they are isomorphic in some category of specifications. We present here some isomorphisms and motivate the use of *construction isomorphisms* to compare *modular* specifications.

Identity. Two specifications are isomorphic if they are identical. This very weak isomorphism is not very interesting.

Structural equivalence. Two specifications are isomorphic if they have been constructed the same way, independently of aliases which may have been defined in the construction process. This isomorphism is slightly less weak than the previous one, but still not very interesting.

Isomorphism in **Spec**. Two specifications are isomorphic if there exists an isomorphism between them in the category of all specifications **Spec**. The difficulty here is first to exhibit the isomorphism between both signatures, and above all to check that it is a *specification morphism*, which is in general undecidable.

Construction isomorphism. Two specifications constructed with colimits from a common category of base specifications $\mathcal{C}_0$ are isomorphic if we can prove it with general properties of colimits. The specifications R_2, R'_2, R_3 and R_4 are isomorphic in this sense. This corresponds to an isomorphism in the category $\mathrm{Term}(\mathcal{C}_0)$ which will be described in section 4. On the one hand, this *construction isomorphism* is not too general in that it reflects the *construction* of the modular specification. On the other hand, it is more general than the structural isomorphism because some irrelevant steps chosen while constructing the modular specification are abstracted. These isomorphisms do not depend on the actual definition of the base specifications, but only on their combination. At last, we show in section 6 that, under certain conditions, we can decide whether two specifications are related by a construction isomorphism.

3 Categorical Setting

This section presents the notion of *equiv-category*, and $\mathrm{DIAGR}(\mathcal{C})$, the equiv-category of diagrams. We mainly restate well known concepts of category theory in the context of equiv-categories. However, the definition of diagram morphism which is proposed here is more general than those usually presented in computer science. The reader not familiar with basic notions of category theory may refer to [1, 24].

3.1 Equiv-categories

An equiv-category is similar to a category, except that equalities between arrows are replaced by *equivalence relations*.

Definition 2 (Equiv-category)
An *equiv-category* $\mathcal{C}$ is a triple $(\mathrm{Obj}(\mathcal{C}), \mathrm{Arr}(\mathcal{C}), \sim)$ such that:

- $\mathrm{Obj}(\mathcal{C})$ is a class of *objects*[2].
- $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, $\mathrm{Arr}(\mathcal{C})(A, B)$ is a set of *arrows from A to B*.
- $\forall A, B \in \mathrm{Obj}(\mathcal{C})$, $\sim$ is a relation on $\mathrm{Arr}(\mathcal{C})(A, B)$.
- $\forall A, B, C \in \mathrm{Obj}(\mathcal{C})$, there is a composition operation
$$\circ : \mathrm{Arr}(\mathcal{C})(B, C) \times \mathrm{Arr}(\mathcal{C})(A, B) \to \mathrm{Arr}(\mathcal{C})(A, C).$$
- $\forall A \in \mathrm{Obj}(\mathcal{C})$, there is an identity arrow $id_A \in \mathrm{Arr}(\mathcal{C})(A, A)$.
- $\forall A, B \in \mathrm{Obj}(\mathcal{C}), \forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$, $f \circ id_A \sim f$ and $id_B \circ f \sim f$.
- $\forall A, B, C, D \in \mathrm{Obj}(\mathcal{C})$,
$\forall f \in \mathrm{Arr}(\mathcal{C})(A, B)$, $\forall g \in \mathrm{Arr}(\mathcal{C})(B, C)$, $\forall h \in \mathrm{Arr}(\mathcal{C})(C, D)$,
$$(h \circ g) \circ f \sim h \circ (g \circ f).$$
- The relation $\sim$ is a congruence i.e.
    - $\sim$ is an equivalence relation, i.e. is reflexive, symmetric and transitive;

[2] Although $\mathrm{Obj}(\mathcal{C})$ may not be a *set*, we talk of "elements" of $\mathrm{Obj}(\mathcal{C})$ and write "$A \in \mathrm{Obj}(\mathcal{C})$".

- $\forall A, B, C \in \mathrm{Obj}(\mathcal{C}),\ \forall f, f' \in \mathrm{Arr}(\mathcal{C})(A, B),\ \forall g, g' \in \mathrm{Arr}(\mathcal{C})(B, C),$
$$f \sim f' \text{ and } g \sim g' \ \Rightarrow\ g \circ f \sim g' \circ f'.$$

Notation: $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ will sometimes be noted $f : A \to B$, when we wish to leave the equiv-category $\mathcal{C}$ implicit.

Definition 3 (Isomorphism) An arrow $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ is an *isomorphism* in an equiv-category $\mathcal{C}$ if there exists $g \in \mathrm{Arr}(\mathcal{C})(B, A)$ such that $g \circ f \sim id_A$ and $f \circ g \sim id_B$. If $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ is an isomorphism, we say that A and B are *isomorphic*, and we note $A \cong B$.

Definition 4 (Equiv-functor) Let $\mathcal{C}$ and $\mathcal{C}'$ be two equiv-categories. An *equiv-functor* F from $\mathcal{C}$ to $\mathcal{C}'$, noted $F : \mathcal{C} \to \mathcal{C}'$, is a map which assigns to each object A of $\mathcal{C}$ an object $F(A)$ of $\mathcal{C}'$, and to each arrow $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ an arrow $F(f) \in \mathrm{Arr}(\mathcal{C}')(F(A), F(B))$, and such that

- $f \sim f' \ \Rightarrow\ F(f) \sim F(f')$;
- $F(id_A) \sim id_{F(A)}$;
- $F(g \circ f) \sim F(g) \circ F(f)$.

Definition 5 (Full equiv-functor) An equiv-functor $F : \mathcal{C} \to \mathcal{C}'$ is *full* if $\forall g \in \mathrm{Arr}(\mathcal{C}')(F(A), F(B)),\ \exists f \in \mathrm{Arr}(\mathcal{C})(A, B)\ ;\ g \sim F(f)$.

Definition 6 (Faithful equiv-functor) An equiv-functor $F : \mathcal{C} \to \mathcal{C}'$ is *faithful* if $\forall f, f' \in \mathrm{Arr}(\mathcal{C})(A, B),\ F(f) \sim F(f') \ \Rightarrow\ f \sim f'$.

Definition 7 (Natural transformation between equiv-functors)
Let $F, G : \mathcal{C} \to \mathcal{C}'$ be two equiv-functors. A natural transformation σ from F to G, noted $\sigma : F \dot{\to} G$, is a map which assigns to every object A of $\mathcal{C}$ an arrow $\sigma_A \in \mathrm{Arr}(\mathcal{C}')(F(A), G(A))$ such that
$$\forall f \in \mathrm{Arr}(\mathcal{C})(A, B),\ G(f) \circ \sigma_A \sim \sigma_B \circ F(f).$$

Let $F, G : \mathcal{C} \to \mathcal{C}'$ be two equiv-functors and $\sigma : F \dot{\to} G$ a natural transformation. If for every $A \in \mathrm{Obj}(\mathcal{C})$, σ_A is an isomorphism, then we say that F and G are *naturally isomorphic*, and we note $F \cong G$. In that case, there is a natural transformation $\sigma^{-1} : G \dot{\to} F$ defined by $(\sigma^{-1})_A = \sigma_A^{-1}$.

Given a graph[3] α^Φ, Vertices(α^Φ) and Edges(α^Φ) respectively denote the set of vertices and the set of edges of α^Φ. An edge a of source m and of target n is noted $a : m \to n$. There is an equiv-category $\mathcal{P}(\alpha^\Phi)$ freely generated over the graph α^Φ which may be defined as follows. The set of objects of $\mathcal{P}(\alpha^\Phi)$ is the set of vertices of α^Φ. Arrows of $\mathcal{P}(\alpha^\Phi)$ are *paths of composable edges* of α^Φ
$$\langle a_1, a_2, \ldots, a_k \rangle$$

[3] By *graph*, we actually mean a directed multigraph, because edges are oriented and there can be more than one edge between any two vertices.

where two edges a_i and a_{i+1} are composable if the target of a_i is equal to the source of a_{i+1}. The congruence relation is the equality on paths. Identity arrows are paths of length 0, noted $\langle\rangle$. The composition is the concatenation of paths:

$$\langle b_1, b_2, \ldots, b_l\rangle \circ \langle a_1, a_2, \ldots, a_k\rangle = \langle a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_l\rangle.$$

Any graph morphism $\sigma^\Phi : \alpha^\Phi \to \beta^\Phi$ from a graph α^Φ to a graph β^Φ uniquely extends to an *equiv-functor* $\mathcal{P}(\sigma^\Phi) : \mathcal{P}(\alpha^\Phi) \to \mathcal{P}(\beta^\Phi)$. $\mathcal{P}(\sigma^\Phi)$ is equal to σ^Φ on objects, and is defined on paths as follows:

$$\begin{array}{lcl} \mathcal{P}(\sigma^\Phi)(\langle\rangle) & = & \langle\rangle\,; \\ \mathcal{P}(\sigma^\Phi)(\langle a_1, a_2, \ldots, a_k\rangle) & = & \langle\sigma^\Phi(a_1), \sigma^\Phi(a_2), \ldots, \sigma^\Phi(a_k)\rangle. \end{array}$$

3.2 Equiv-category of diagrams

A *diagram* $\overline{\alpha}$ over an equiv-category $\mathcal{C}$ consists of a graph α^Φ whose vertices m are labeled by objects $\alpha(m)$ of $\mathcal{C}$ and whose edges $a : m \to n$ are labeled by arrows $\alpha(a) : \alpha(m) \to \alpha(n)$ of $\mathcal{C}$. Formally:

Definition 8 (Diagram over an equiv-category)
A diagram $\overline{\alpha}$ over an equiv-category $\mathcal{C}$ is a couple $\overline{\alpha} = (\alpha^\Phi,\ \alpha : \mathcal{P}(\alpha^\Phi) \to \mathcal{C})$, where α^Φ is a graph and $\alpha : \mathcal{P}(\alpha^\Phi) \to \mathcal{C}$ is an equiv-functor.

We say that the diagram $\overline{\alpha}$ is *based* on the graph α^Φ, or that α^Φ is the *underlying* graph of $\overline{\alpha}$. A diagram $\overline{\alpha}$ is *finite* when its underlying graph is finite.

Examples of diagrams

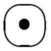
1. The empty diagram, noted $\bigcirc$, is the only diagram based on the empty graph.

2. Let A be an object of $\mathcal{C}$. There is a one point diagram

   $$I(A) = (1^\Phi,\ I^A : \mathcal{P}(1^\Phi) \to \mathcal{C}),$$

   where

   - 1^Φ is a graph which has only one vertex, noted $*$;
   - the equiv-functor I^A is defined by $I^A(*) = A$.



graph 1^Φ

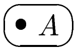
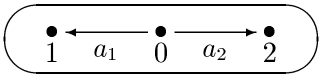


diagram $I(A)$

3. Let π^Φ be the graph consisting of three vertices 0, 1, 2, and two edges $a_1 : 0 \to 1$, $a_2 : 0 \to 2$. A *pushout diagram* is a diagram based on π^Φ. If $A, B, C \in \mathrm{Obj}(\mathcal{C})$, $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ and $g \in \mathrm{Arr}(\mathcal{C})(A, C)$, then there is a *pushout diagram*

   $$\mathcal{PushDiagr}(A, B, C, f, g) = (\pi^\Phi,\ \pi : \mathcal{P}(\pi^\Phi) \to \mathcal{C})$$

   defined by $\pi(0) = A,\ \pi(1) = B,\ \pi(2) = C,\ \pi(a_1) = f,\ \pi(a_2) = g.$



graph π^Φ

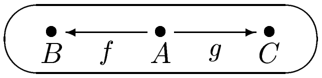


diagram $\mathcal{P}ush\mathcal{D}iagr(A,B,C,f,g)$

We now define *diagram morphisms.* We could consider couples

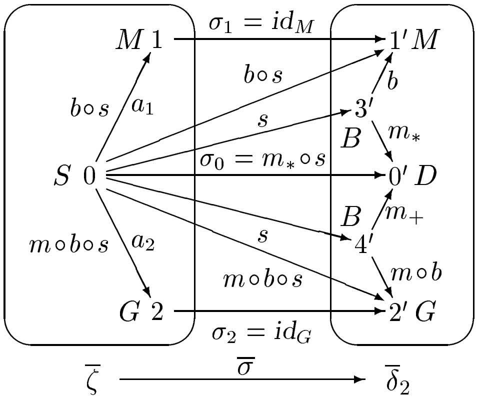
$$\overline{\sigma}:\overline{\alpha}\to\overline{\beta}=(\sigma^\Phi:\alpha^\Phi\to\beta^\Phi,\ \sigma:\alpha\mathrel{\dot{\to}}\beta\circ\mathcal{P}(\sigma^\Phi))$$

where σ^Φ is a graph morphism and σ a natural transformation. This definition is presented in [34] (it is the flatten category $\mathbf{Func}(\mathcal{C}_0)$, page 244, example 4), and in a dual form in [24] (it is the super-comma category, page 111, exercise 5.b). This definition is not general enough for our purpose because the resulting category is not, in the general case, finitely cocomplete. It indeed does not contain enough arrows, and therefore some specification morphisms have no corresponding diagram morphism.

To come back to the example presented in section 2, there is indeed a term

$$\mathsf{T}_{\mathsf{up}}=\mathsf{up}(S,M,G,R_2,b\circ s,m\circ b\circ s,\mathsf{k}_1(R_2)\circ\mathsf{k}_1(MD),\mathsf{k}_2(R_2))$$

from P to R_2, which should correspond to an arrow from the diagram $\overline{\zeta}$ to the diagram $\overline{\delta_2}$. However, there is no diagram morphism with the definition above.



For this reason, instead of considering a graph morphism $\sigma^\Phi:\alpha^\Phi\to\beta^\Phi$, we need to consider a *generalized graph morphism* $\sigma^\Phi:\alpha^\Phi\rightsquigarrow\beta^\Phi$, which assigns to each edge of α^Φ a *zigzag* of β^Φ. Instead of considering a natural transformation $\sigma:\alpha\mathrel{\dot{\to}}\beta\circ\mathcal{P}(\sigma^\Phi)$, we need to consider a *generalized natural transformation* $\sigma:\alpha\mathrel{\dot{\rightsquigarrow}}\beta\circ\mathcal{P}(\sigma^\Phi)$ [27].

Definition 9 (Zigzag) A *zigzag* Z of a graph α^Φ is a triple (k,Z_V,Z_E) where

- k is a natural, called the *length* of Z;
- Z_V is a $(k+1)$-uple $(n_0,n_1,\ldots,n_k)$ of vertices of α^Φ;
- Z_E is a k-uple $(a_0,a_1,\ldots,a_{k-1})$ of edges of α^Φ, such that $\forall i, 0\le i\le k-1$, either $a_i:n_i\to n_{i+1}$ (i.e. a_i is an edge of source n_i and of target n_{i+1}), or $a_i:n_{i+1}\to n_i$ (i.e. a_i is an edge of source n_{i+1} and of target n_i).

A zigzag is noted $Z:n_0\rightsquigarrow n_k$, or, more pictorially,

$$Z\;=\;n_0\xrightarrow{a_0}n_1\xleftarrow{a_1}n_2\xleftarrow{a_2}n_3\cdots n_{k-1}\xrightarrow{a_{k-1}}n_k,$$

where an arbitrary orientation is chosen for each edge a_i.

For every vertex n of α^Φ, there is an *empty zigzag* $0_n : n \rightsquigarrow n$. We get a graph $\mathrm{Zigzag}(\alpha^\Phi)$, with the same vertices as α^Φ, and with edges the zigzags of α^Φ.

Definition 10 (Generalized graph morphism)
A *generalized graph morphism* σ^Φ from α^Φ to β^Φ, noted $\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$, is a graph morphism from α^Φ to $\mathrm{Zigzag}(\beta^\Phi)$.

We can compose generalized graph morphisms by joining zigzags.

Definition 11 (Connection relation) Let $\overline{\delta} = (\delta^\Phi,\ \delta : \mathcal{P}(\delta^\Phi) \to \mathcal{C})$ be a diagram, and n, n' two vertices of δ^Φ. Two arrows $u : A \to \delta(n)$ and $v : A \to \delta(n')$ of $\mathcal{C}$ are said to be *connected by the diagram* $\overline{\delta}$ if and only if there exist a zigzag $Z : n_0 \rightsquigarrow n_k$ of δ^Φ with $n = n_0$ and $n' = n_k$

$$Z \;=\; n{=}n_0 \xrightarrow{a_0} n_1 \xleftarrow{a_1} \cdots n_{k-1} \xrightarrow{a_{k-1}} n_k{=}n',$$

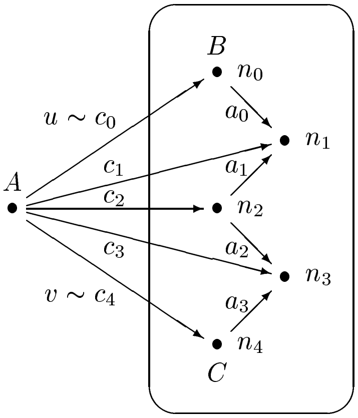
and a set of arrows

$$\{c_i : A \to \delta(n_i)\,;\ i \in \{0, \ldots, k\}\},$$

such that $u \sim c_0$, $v \sim c_k$ and $\forall i \in \{0, \ldots k-1\}$,

- $\delta(a_i) \circ c_i \sim c_{i+1}$, if $a_i : n_i \to n_{i+1}$;
- $\delta(a_i) \circ c_{i+1} \sim c_i$, if $a_i : n_{i+1} \to n_i$.

We note $u \sim_{\overline{\delta}} v$, or $u \sim_{\overline{\delta}} v\ [Z]$ if we want to specify the zigzag Z.



$\overline{\delta}$

$u \sim_{\overline{\delta}} v\ \ [Z : n_0 \rightsquigarrow n_4]$

Definition 12 (Diagram morphism) Let $\mathcal{C}$ be an equiv-category, $\overline{\alpha}$ and $\overline{\beta}$ be two diagrams over $\mathcal{C}$. A *diagram morphism* $\overline{\sigma}$ from $\overline{\alpha}$ to $\overline{\beta}$, noted $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$, is a couple

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

where

- $\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$ is a generalized graph morphism;
- $\sigma : \alpha \overset{\cdot}{\rightsquigarrow} \beta \circ \mathcal{P}(\sigma^\Phi)$ is a generalized natural transformation, i.e. a set of arrows $\{\sigma_n : \alpha(n) \to \beta(\sigma^\Phi(n)),\ \forall n \in \mathrm{Vertices}(\alpha^\Phi)\}$ such that
$$\forall a : m \to n \in \mathrm{Edges}(\alpha^\Phi),\ \sigma_n \circ \alpha(a) \sim_{\overline{\beta}} \sigma_m\ [\sigma^\Phi(a)].$$

Note: if σ^Φ is a graph morphism, then $\mathcal{P}(\sigma^\Phi) : \mathcal{P}(\alpha^\Phi) \to \mathcal{P}(\beta^\Phi)$ is a functor and $\sigma : \alpha \overset{\cdot}{\to} \beta \circ \mathcal{P}(\sigma^\Phi)$ is a natural transformation.

Examples of diagram morphisms

1. For every diagram $\overline{\alpha}$, there is a diagram morphism
$$\overline{\mathcal{I}d_{\overline{\alpha}}} : \overline{\alpha} \to \overline{\alpha} = (id_{\alpha^\Phi} : \alpha^\Phi \rightsquigarrow \alpha^\Phi,\ \mathcal{I}d_{\overline{\alpha}} : \alpha \overset{\cdot}{\rightsquigarrow} \alpha)$$
where id_{α^Φ} is the identity (generalized) graph morphism and $\mathcal{I}d_{\overline{\alpha}} : \alpha \overset{\cdot}{\rightsquigarrow} \alpha$ is a (generalized) natural transformation defined by $(\mathcal{I}d_{\overline{\alpha}})_n = id_{\alpha(n)}$.

2. For every diagram $\overline{\alpha}$, there is a (unique) diagram morphism from the empty diagram to $\overline{\alpha}$, noted $\overline{\mathcal{J}_{\overline{\alpha}}} : \bigcirc \to \overline{\alpha}$.

3. We can define a diagram morphism $\overline{\sigma} : \overline{\zeta} \to \overline{\delta_2}$ corresponding to the term T_{up}. This arrow consists of a generalized graph morphism $\sigma^\Phi : \zeta^\Phi \rightsquigarrow \delta_2^\Phi$ and a generalized natural transformation $\sigma : \zeta \overset{\cdot}{\rightsquigarrow} \delta_2 \circ \mathcal{P}(\sigma^\Phi)$.

   The generalized graph morphism σ^Φ may be defined as follows:

   $$\begin{array}{l} \sigma^\Phi(0) = 0',\ \sigma^\Phi(1) = 1',\ \sigma^\Phi(2) = 2', \\ \sigma^\Phi(a_1) = 0' \xleftarrow{m_*} 3' \xrightarrow{b} 1',\ \sigma^\Phi(a_2) = 0' \xleftarrow{m_+} 4' \xrightarrow{m \circ b} 2'. \end{array}$$

   The generalized natural transformation $\sigma : \zeta \overset{\cdot}{\rightsquigarrow} \delta_2 \circ \mathcal{P}(\sigma^\Phi)$ is defined by

   $$\sigma_0 = m_* \circ s,\ \sigma_1 = id_M,\ \sigma_2 = id_G.$$

   We indeed defined a diagram morphism $\overline{\sigma}$ because $m_+ \circ s \sim m_* \circ s$.

   Another diagram morphism $\overline{\tau} : \overline{\zeta} \to \overline{\delta_2}$, which also corresponds to the term T_{up}, may be defined as follows.

   The generalized graph morphism τ^Φ is

   $$\begin{array}{l} \tau^\Phi(0) = 3',\ \tau^\Phi(1) = 1',\ \tau^\Phi(2) = 2', \\ \tau^\Phi(a_1) = 3' \xrightarrow{b} 1',\ \tau^\Phi(a_2) = 3' \xrightarrow{m_*} 0' \xleftarrow{m_+} 4' \xrightarrow{m \circ b} 2'. \end{array}$$

   The generalized natural transformation $\tau : \zeta \overset{\cdot}{\rightsquigarrow} \delta_2 \circ \mathcal{P}(\tau^\Phi)$ is defined by

   $$\tau_0 = s,\ \tau_1 = id_M,\ \tau_2 = id_G.$$

To get an *equiv-category* of diagrams, it remains to define a congruence on diagram morphisms. Intuitively, two diagram morphisms $\overline{\sigma}, \overline{\tau} : \overline{\alpha} \to \overline{\beta}$ are equivalent if they correspond to the same colimiting arrow from $\overline{\alpha}$ to $\overline{\beta}$.

Definition 13 (Equivalence $\approx$) Let $\overline{\sigma}, \overline{\tau} : \overline{\alpha} \to \overline{\beta}$ be two diagram morphisms. By definition, $\overline{\sigma} \approx \overline{\tau}$ if $\forall n \in \text{Vertices}(\alpha^\Phi),\ \sigma_n \sim_{\overline{\beta}} \tau_n$.

For instance, in the example above, the diagram morphisms $\overline{\sigma}, \overline{\tau} : \overline{\zeta} \to \overline{\delta_2}$, which correspond to the same specification morphism T_{up}, are equivalent.

Theorem 14 DIAGR($\mathcal{C}$) *is an equiv-category, which has diagrams as objects, diagram morphisms as arrows and $\approx$ as congruence relation.*

Proof. First, we must define the composition of diagram morphisms. Let $\overline{\alpha}$, $\overline{\beta}$ and $\overline{\gamma}$ be three diagrams. Let

$$\begin{array}{l} \overline{\sigma} : \overline{\alpha} \to \overline{\beta} = (\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi,\ \sigma : \alpha \overset{\cdot}{\rightsquigarrow} \beta \circ \mathcal{P}(\sigma^\Phi)) \\ \overline{\tau} : \overline{\beta} \to \overline{\gamma} = (\tau^\Phi : \beta^\Phi \rightsquigarrow \gamma^\Phi,\ \tau : \beta \overset{\cdot}{\rightsquigarrow} \gamma \circ \mathcal{P}(\tau^\Phi)) \end{array}$$

be two diagram morphisms. The composition $\overline{\kappa} = \overline{\tau} \circ \overline{\sigma}$ of $\overline{\sigma}$ and $\overline{\tau}$ is the couple

$$\overline{\kappa} : \overline{\alpha} \to \overline{\gamma} = (\kappa^\Phi : \alpha^\Phi \rightsquigarrow \gamma^\Phi,\ \kappa : \alpha \overset{\cdot}{\rightsquigarrow} \gamma \circ \mathcal{P}(\kappa^\Phi))$$

where

- $\kappa^\Phi = \tau^\Phi \circ \sigma^\Phi$;
- the generalized natural transformation $\kappa : \alpha \rightsquigarrow\!\!\!\cdot\!\!\to \gamma \circ \mathcal{P}(\kappa^\Phi)$ is defined by: $\forall n \in \text{Vertices}(\alpha^\Phi),\ \kappa_n = \tau_{\sigma^\Phi(n)} \circ \sigma_n$.

One has to check that κ is indeed a general natural transformation and that the composition is associative.

For every diagram $\overline{\alpha}$, the diagram morphism $\overline{\mathcal{I}d}_{\overline{\alpha}} : \overline{\alpha} \to \overline{\alpha}$ is an identity.

It then remains to show that $\approx$ is a congruence. $\approx$ is an equivalence relation, because $\sim_{\overline{\beta}}$ is an equivalence relation. To prove that it is a congruence, we show that given two arrows $u \in \text{Arr}(\mathcal{C})(A, \beta(m))$ and $v \in \text{Arr}(\mathcal{C})(A, \beta(n))$, we have

- $\forall w \in \text{Arr}(\mathcal{C})(A', A),\ u \sim_{\overline{\beta}} v\ [Z] \ \Rightarrow\ u \circ w \sim_{\overline{\beta}} v \circ w\ [Z]$;
- $\forall \overline{\tau} : \overline{\beta} \to \overline{\gamma},\ u \sim_{\overline{\beta}} v\ [Z] \ \Rightarrow\ \tau_m \circ u \sim_{\overline{\gamma}} \tau_n \circ v\ [\tau^\Phi(Z)]$.

□

Definition 15 (Equiv-functor $I : \mathcal{C} \to \text{DIAGR}(\mathcal{C})$)
We define an equiv-functor $I : \mathcal{C} \to \text{DIAGR}(\mathcal{C})$ as follows. I assigns to each object A of $\mathcal{C}$ the diagram $I(A)$, and to each arrow $f \in \text{Arr}(\mathcal{C})(A, B)$ the diagram morphism $I(f) = (id_{1^\Phi} : 1^\Phi \rightsquigarrow 1^\Phi,\ I^f : I^A \rightsquigarrow\!\!\!\cdot\!\!\to I^B)$ defined by $I^f_* = f$.

$$\begin{array}{ccccc}
I & : & \mathcal{C} & \to & \text{DIAGR}(\mathcal{C}) \\
 & & A & \mapsto & \boxed{\bullet\ A} \\
 & & f : A \to B & \mapsto & \boxed{A\ \bullet} \xrightarrow{f} \boxed{\bullet\ B}
\end{array}$$

We can check that I is full and faithful.

Definition 16 (Cone) Let $\overline{\alpha}$ be a diagram over an equiv-category $\mathcal{C}$ and C be an object of $\mathcal{C}$. A *cone from* $\overline{\alpha}$ is a diagram morphism $\overline{\lambda} : \overline{\alpha} \to I(C)$.

Remark 17 $\overline{\lambda} : \overline{\alpha} \to I(C)$ is a diagram morphism if and only if

$$\forall a : m \to n \in \text{Edges}(\alpha^\Phi),\ \lambda_n \circ \alpha(a) \sim \lambda_m.$$

Definition 18 (Equiv-colimiting cone) Let $\overline{\alpha}$ be a diagram over an equiv-category $\mathcal{C}$. A cone $\overline{\lambda} : \overline{\alpha} \to I(C)$ from $\overline{\alpha}$ is an *equiv-colimiting cone* if for any cone $\overline{\mu} : \overline{\alpha} \to I(D)$ from $\overline{\alpha}$, there exists an arrow $\psi \in \text{Arr}(\mathcal{C})(C, D)$, unique up to equivalence, such that $I(\psi) \circ \overline{\lambda} \approx \overline{\mu}$.

The object C is called an *equiv-colimit* of the diagram $\overline{\alpha}$ and is noted $\text{Colim}\,\overline{\alpha}$. The arrow ψ is called a *mediating arrow* from $\overline{\lambda}$ to $\overline{\mu}$. A diagram may have several equiv-colimits which are then isomorphic. Given a diagram $\overline{\alpha}$ which has an equiv-colimiting cone, one can *choose an equiv-colimit* of $\overline{\alpha}$. This means

1. choose an equiv-colimiting cone $\overline{\lambda} : \overline{\alpha} \to I(C)$;
2. for every cone $\overline{\mu} : \overline{\alpha} \to I(D)$, choose a mediating arrow $\psi : C \to D$ from $\overline{\lambda}$ to $\overline{\mu}$ in $\mathcal{C}$.

Examples of equiv-colimits

- Given an object A of $\mathcal{C}$, any object isomorphic to A is an equiv-colimit of the diagram $I(A)$.
- An *equiv-initial object* is an equiv-colimit of the empty diagram $\bigcirc$.

  If an equiv-category $\mathcal{C}$ has a chosen equiv-initial object, this one is noted $\emptyset^{\mathcal{C}}$. The chosen mediating arrow from $\emptyset^{\mathcal{C}}$ to any object A is noted $j_A^{\mathcal{C}} : \emptyset^{\mathcal{C}} \to A$.
- An *equiv-pushout* is an equiv-colimit of a pushout diagram. For instance, the equiv-colimit of $\mathcal{PushDiagr}(A, B, C, f, g)$ consists of a triple (P, k_1, k_2) where P is an object of $\mathcal{C}$, $k_1 : B \to P$ and $k_2 : C \to P$ are two arrows of $\mathcal{C}$, such that

  - $k_1 \circ f \sim k_2 \circ g$;
  - $\forall D \in \mathrm{Obj}(\mathcal{C}), f' \in \mathrm{Arr}(\mathcal{C})(B, D), g' \in \mathrm{Arr}(\mathcal{C})(C, D)$ such that $f' \circ f \sim g' \circ g$, there exists an arrow $up : P \to D$, unique up to equivalence, such that $up \circ k_1 \sim f'$ and $up \circ k_2 \sim g'$.

  If $\mathcal{PushDiagr}(A, B, C, f, g)$ has a *chosen* equiv-colimit, P, k_1 and k_2 are respectively noted

  $push^{\mathcal{C}}(A, B, C, f, g)$
  $k_1^{\mathcal{C}}(A, B, C, f, g) : B \to push^{\mathcal{C}}(A, B, C, f, g)$
  $k_2^{\mathcal{C}}(A, B, C, f, g) : C \to push^{\mathcal{C}}(A, B, C, f, g)$.

  For all $f' \in \mathrm{Arr}(\mathcal{C})(B, D)$ and $g' \in \mathrm{Arr}(\mathcal{C})(C, D)$ such that $f' \circ f \sim g' \circ g$, the chosen mediating arrow from P to D is noted

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

Definition 19 (Finitely cocomplete equiv-category) An equiv-category $\mathcal{C}$ is *finitely cocomplete* if every finite diagram over $\mathcal{C}$ has an equiv-colimit.

Let $\mathcal{C}$ be a finitely cocomplete equiv-category. Thus, every diagram $\overline{\alpha}$ has an equiv-colimiting cone, noted $\overline{\eta}_{\overline{\alpha}} : \overline{\alpha} \to I(\mathrm{Colim}\,\overline{\alpha})$. We show that we can extend the map Colim to arrows so that $\mathrm{Colim} : \mathrm{DIAGR}(\mathcal{C}) \to \mathcal{C}$ is an equiv-functor.

Let $\overline{\alpha}$ and $\overline{\beta}$ be two diagrams, with equiv-colimiting cones $\overline{\eta}_{\overline{\alpha}} : \overline{\alpha} \to I(\mathrm{Colim}\,\overline{\alpha})$ and $\overline{\eta}_{\overline{\beta}} : \overline{\beta} \to I(\mathrm{Colim}\,\overline{\beta})$. Let $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ be a diagram morphism. We have a cone $\overline{\eta}_{\overline{\beta}} \circ \overline{\sigma} : \overline{\alpha} \to I(\mathrm{Colim}\,\overline{\beta})$. As $\overline{\eta}_{\overline{\alpha}} : \overline{\alpha} \to I(\mathrm{Colim}\,\overline{\alpha})$ is an equiv-colimiting cone from $\overline{\alpha}$, there exists an arrow

$$\mathrm{Colim}\,\overline{\sigma} : \mathrm{Colim}\,\overline{\alpha} \to \mathrm{Colim}\,\overline{\beta},$$

unique up to equivalence, such that $I(\mathrm{Colim}\,\overline{\sigma}) \circ \overline{\eta}_{\overline{\alpha}} \approx \overline{\eta}_{\overline{\beta}} \circ \overline{\sigma}$.

Proposition 20 *Let $\mathcal{C}$ be a finitely cocomplete equiv-category.*

1. $\mathrm{Colim} : \mathrm{DIAGR}(\mathcal{C}) \to \mathcal{C}$ *is an equiv-functor.*
2. *There is a natural transformation $\overline{\eta} : Id_{\mathrm{DIAGR}(\mathcal{C})} \dot{\to} I \circ \mathrm{Colim}$.*
3. *Between the equiv-functors* Colim *and I, there is an adjunction* $(\mathrm{Colim} \dashv I)$ *whose unit is $\overline{\eta}$. This means that for each object B of $\mathcal{C}$ and arrow $\overline{\mu} : \overline{\alpha} \to I(B)$ of* $\mathrm{DIAGR}(\mathcal{C})$, *there exists an arrow ψ : $\mathrm{Colim}\,\overline{\alpha} \to B$ of $\mathcal{C}$, unique up to equivalence, such that $I(\psi) \circ \overline{\eta}_{\overline{\alpha}} \approx \overline{\mu}$.*
4. *The counit $\epsilon : \mathrm{Colim} \circ I \dot{\to} Id_{\mathcal{C}}$ of the adjunction* $(\mathrm{Colim} \dashv I)$ *is a natural isomorphism.*

Proof.

1. We just show that Colim is compatible with relations.

$$\begin{array}{rll} \overline{\sigma} \approx \overline{\tau} & \Rightarrow \ \overline{\eta}_{\overline{\beta}} \circ \overline{\sigma} \approx \overline{\eta}_{\overline{\beta}} \circ \overline{\tau} & (\approx \text{ is a congruence}) \\ & \Rightarrow \ I(\mathrm{Colim}\,\overline{\sigma}) \circ \overline{\eta}_{\overline{\alpha}} \approx I(\mathrm{Colim}\,\overline{\tau}) \circ \overline{\eta}_{\overline{\alpha}} & (\text{definition of Colim}) \\ & \Rightarrow \ \mathrm{Colim}\,\overline{\sigma} \sim \mathrm{Colim}\,\overline{\tau} & (\overline{\eta}_{\overline{\alpha}} \text{ equiv-colimiting cone}) \end{array}$$

2. Immediate by the definition of $\mathrm{Colim}\,\overline{\sigma}$.

3. $\overline{\eta}_{\overline{\alpha}} : \overline{\alpha} \rightarrow I(\mathrm{Colim}\,\overline{\alpha})$ is an equiv-colimiting cone from $\overline{\alpha}$.

4. Using the law $I\epsilon \cdot \overline{\eta} I \approx id_I$ which relates the unit and the counit of the adjunction, we show that for all object B of $\mathcal{C}$, $\epsilon_B \circ (\eta_{I(B)})_* \sim id_B$ and $(\eta_{I(B)})_* \circ \epsilon_B \sim id_{\mathrm{Colim}\, I(B)}$.

□

3.3 Preservation of equiv-colimits

In this paragraph, we define the image of a diagram and of a diagram morphism over $\mathcal{C}$ by an equiv-functor $F : \mathcal{C} \rightarrow \mathcal{C}'$.

Definition 21 (Image of a diagram) Let $\overline{\alpha} = (\alpha^{\Phi},\ \alpha : \mathcal{P}(\alpha^{\Phi}) \rightarrow \mathcal{C})$ be a diagram over $\mathcal{C}$. The *image* of $\overline{\alpha}$ by F is the diagram over $\mathcal{C}'$

$$F \circ \overline{\alpha} = (\alpha^{\Phi},\ F \circ \alpha : \mathcal{P}(\alpha^{\Phi}) \rightarrow \mathcal{C}').$$

Definition 22 (Image of a diagram morphism)
Let $\overline{\sigma} : \overline{\alpha} \rightarrow \overline{\beta} = (\sigma^{\Phi} : \alpha^{\Phi} \rightsquigarrow \beta^{\Phi},\ \sigma : \alpha \overset{\cdot}{\rightsquigarrow} \beta \circ \mathcal{P}(\sigma^{\Phi}))$ be a diagram morphism over $\mathcal{C}$. The *image* of $\overline{\sigma}$ by F is the diagram morphism over $\mathcal{C}'$

$$F\overline{\sigma} : F \circ \overline{\alpha} \rightarrow F \circ \overline{\beta} = (\sigma^{\Phi} : \alpha^{\Phi} \rightsquigarrow \beta^{\Phi},\ F\sigma : F \circ \alpha \overset{\cdot}{\rightsquigarrow} F \circ \beta \circ \mathcal{P}(\sigma^{\Phi})),$$

where the generalized natural transformation $F\sigma$ is defined by

$$\forall n \in \mathrm{Vertices}(\alpha^{\Phi}),\ (F\sigma)_n = F(\sigma_n).$$

Lemma 23 *The map*

$$\begin{array}{rcccc} \mathrm{DIAGR}(F) & : & \mathrm{DIAGR}(\mathcal{C}) & \rightarrow & \mathrm{DIAGR}(\mathcal{C}') \\ & & \overline{\alpha} & \mapsto & F \circ \overline{\alpha} \\ & & \overline{\sigma} & \mapsto & F\overline{\sigma} \end{array}$$

is an equiv-functor such that $I \circ F = \mathrm{DIAGR}(F) \circ I$.

An equiv-functor *preserves* a (chosen) equiv-colimit when the image of a (chosen) equiv-colimit is a (chosen) equiv-colimit.

Definition 24 (Equiv-functor preserving an equiv-colimit) Let $\overline{\alpha}$ be a diagram over $\mathcal{C}$, with an equiv-colimiting cone $\overline{\lambda} : \overline{\alpha} \rightarrow I(C)$. The equiv-functor F *preserves* the equiv-colimit of $\overline{\alpha}$ if $F\overline{\lambda}$ is an equiv-colimiting cone from $F \circ \overline{\alpha}$.

Lemma 25 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two finitely cocomplete equiv-categories. If an equiv-functor $F : \mathcal{C} \to \mathcal{C}'$ preserves all equiv-colimits, then there is a natural isomorphism*

$$\mathrm{Colim} \circ \mathrm{DIAGR}(F) \cong F \circ \mathrm{Colim}.$$

Definition 26 (Equiv-functor preserving a chosen equiv-colimit)
Let $\overline{\alpha}$ be a diagram over $\mathcal{C}$ with a chosen equiv-colimit, i.e. a chosen equiv-colimiting cone $\overline{\lambda} : \overline{\alpha} \to I(C)$ and a chosen mediating arrow $\psi : C \to D$ for any cone $\overline{\mu} : \overline{\alpha} \to I(D)$. The equiv-functor F *preserves the chosen equiv-colimit* of $\overline{\alpha}$ if $F\overline{\lambda}$ is the chosen equiv-colimiting cone from $F \circ \overline{\alpha}$ and $F(\psi)$ is the chosen mediating arrow of the cone $F\overline{\mu}$.

Let us consider the pushout diagram $\mathcal{P}ush\mathcal{D}iagr(A, B, C, f, g)$. Then, F preserves its chosen equiv-colimit if

$$\begin{aligned}
&F(push^{\mathcal{C}}(A, B, C, f, g)) = push^{\mathcal{C}'}(F(A), F(B), F(C), F(f), F(g))\\
&F(k_1^{\mathcal{C}}(A, B, C, f, g)) = k_1^{\mathcal{C}'}(F(A), F(B), F(C), F(f), F(g))\\
&F(k_2^{\mathcal{C}}(A, B, C, f, g)) = k_2^{\mathcal{C}'}(F(A), F(B), F(C), F(f), F(g))\\
&F(up^{\mathcal{C}}(A, B, C, D, f, g, f', g'))\\
&\qquad = up^{\mathcal{C}'}(F(A), F(B), F(C), F(D), F(f), F(g), F(f'), F(g')).
\end{aligned}$$

F preserves the chosen initial object $\emptyset^{\mathcal{C}}$ of $\mathcal{C}$ if

$$\begin{aligned}
&F(\emptyset^{\mathcal{C}}) = \emptyset^{\mathcal{C}'}\\
&F(j_A^{\mathcal{C}}) = j_{F(A)}^{\mathcal{C}'}.
\end{aligned}$$

3.4 Flattening

In this paragraph, we show that the equiv-category of diagrams $\mathrm{DIAGR}(\mathcal{C})$ is finitely cocomplete. In other words, every diagram over $\mathrm{DIAGR}(\mathcal{C})$, i.e. every object of $\mathrm{DIAGR}^2(\mathcal{C})$ has an equiv-colimit. An object of $\mathrm{DIAGR}^2(\mathcal{C})$

$$\overline{\overline{\Delta}} = (\Delta^{\Phi},\ \Delta : \mathcal{P}(\Delta^{\Phi}) \to \mathrm{DIAGR}(\mathcal{C}))$$

is a graph Δ^{Φ} whose vertices N are labeled by diagrams $\overline{\Delta(N)}$ (which are objects of the equiv-category $\mathrm{DIAGR}(\mathcal{C})$), and whose edges $A : N \to N'$ are labeled by diagram morphisms $\overline{\Delta(A)} : \overline{\Delta(N)} \to \overline{\Delta(N')}$ (which are arrows of $\mathrm{DIAGR}(\mathcal{C})$). We will show that an equiv-colimit of $\overline{\overline{\Delta}}$ may be computed by *flattening* this diagram. Intuitively, flattening $\overline{\overline{\Delta}}$ consists in considering the union of all diagrams $\overline{\Delta(N)}$, and in transforming every arrow of $\mathrm{DIAGR}(\mathcal{C})$ into a set of arrows of $\mathcal{C}$.

The congruence relation in $\mathrm{DIAGR}^2(\mathcal{C})$ will be noted $\approxeq$.

Definition 27 (Flattening $\mathrm{Apl} : \mathrm{DIAGR}^2(\mathcal{C}) \to \mathrm{DIAGR}(\mathcal{C})$)
Flattening is a map which assigns to each object

$$\overline{\overline{\Delta}} = (\Delta^{\Phi},\ \Delta : \mathcal{P}(\Delta^{\Phi}) \to \mathrm{DIAGR}(\mathcal{C}))$$

of $\mathrm{DIAGR}^2(\mathcal{C})$ an object $\mathrm{Apl}\,\overline{\overline{\Delta}} = \overline{\delta} = (\delta^{\Phi},\ \delta : \mathcal{P}(\delta^{\Phi}) \to \mathcal{C})$ of $\mathrm{DIAGR}(\mathcal{C})$ as follows.

- δ^{Φ} is a graph defined by its set of vertices and its set of edges.

$$
\begin{array}{lll}
\mathrm{Vertices}(\delta^\Phi) & = \{\ (N,n)\ ;\ N \in \mathrm{Vertices}(\Delta^\Phi),\ n \in \mathrm{Vertices}(\Delta(N)^\Phi)\ \} & \\
 & & \\
\mathrm{Edges}(\delta^\Phi) & = \{\ (N,a) : (N,n) \to (N,n')\ ; & \\
 & \qquad\qquad N \in \mathrm{Vertices}(\Delta^\Phi),\ n,n' \in \mathrm{Vertices}(\Delta(N)^\Phi), & \\
 & \qquad\qquad a : n \to n' \in \mathrm{Edges}(\Delta(N)^\Phi)\ \} & \\
 & \cup\ \{\ (A,n) : (N,n) \to (N',\Delta(A)^\Phi(n))\ ; & \\
 & \qquad\qquad N,N' \in \mathrm{Vertices}(\Delta^\Phi),\ A : N \to N' \in \mathrm{Edges}(\Delta^\Phi), & \\
 & \qquad\qquad n \in \mathrm{Vertices}(\Delta(N)^\Phi)\ \} &
\end{array}
$$

- $\delta : \mathcal{P}(\delta^\Phi) \to \mathcal{C}$ is an equiv-functor defined on vertices and edges of δ^Φ as follows.
  - Action on vertices: $\delta(N,n) = \Delta(N)(n)$.
  - Action on edges: $\delta(N,a) = \Delta(N)(a)$
    $\delta(A,n) = \Delta(A)_n$.

For each vertex N of Δ^Φ, we define an arrow $\overline{\mathcal{K}_N} : \overline{\Delta(N)} \to \overline{\delta}$ in $\mathrm{DIAGR}(\mathcal{C})$

$$\overline{\mathcal{K}_N} : \overline{\Delta(N)} \to \overline{\delta} = (\mathcal{K}_N^\Phi : \Delta(N)^\Phi \rightsquigarrow \delta^\Phi,\ \mathcal{K}_N : \Delta(N) \overset{.}{\rightsquigarrow} \delta \circ \mathcal{P}(\mathcal{K}_N^\Phi)).$$

- The generalized graph morphism $\mathcal{K}_N^\Phi : \Delta(N)^\Phi \rightsquigarrow \delta^\Phi$ is defined by

$$
\begin{array}{cccc}
\mathcal{K}_N^\Phi & : & \Delta(N)^\Phi & \rightsquigarrow & \delta^\Phi \\
 & & n & \mapsto & (N,n) \\
 & & a : n \to n' & \mapsto & (N,a) : (N,n) \to (N,n').
\end{array}
$$

  $\mathcal{K}_N^\Phi$ is actually a graph morphism because each edge of $\Delta(N)^\Phi$ is assigned to an *edge* of δ^Φ (and not to any zigzag).

- The (generalized) natural transformation $\mathcal{K}_N : \Delta(N) \overset{.}{\rightsquigarrow} \delta \circ \mathcal{P}(\mathcal{K}_N^\Phi)$ assigns to each vertex n of $\Delta(N)^\Phi$ the arrow of $\mathcal{C}$

$$(\mathcal{K}_N)_n = id_{\Delta(N)(n)} = id_{\delta(N,n)}.$$

Note. In the following, we will think of $(\mathcal{K}_N)_n$ as an isomorphism from $\Delta(N)(n)$ to $\delta(N,n)$ which is consistent with the fact that $\overline{\mathcal{K}_N}$ is a diagram morphism from $\overline{\Delta(N)}$ to $\overline{\delta}$.

Lemma 28 *The set of arrows* $\{\overline{\mathcal{K}_N} : \overline{\Delta(N)} \to \overline{\delta};\ \forall N \in \mathrm{Vertices}(\Delta^\Phi)\}$ *defines an arrow* $\overline{\overline{\mathcal{K}}} : \overline{\overline{\Delta}} \to I(\overline{\delta})$ *in* $\mathrm{DIAGR}^2(\mathcal{C})$.

Proof. We need to show that

$$
\begin{array}{l}
\forall A : N \to N' \in \mathrm{Edges}(\Delta^\Phi),\ \overline{\mathcal{K}_{N'}} \circ \overline{\Delta(A)} \approx \overline{\mathcal{K}_N} \\
\quad \Leftrightarrow \quad \forall n \in \mathrm{Vertices}(\Delta(N)^\Phi),\ (\mathcal{K}_{N'})_{\Delta(A)^\Phi(n)} \circ \Delta(A)_n \sim_{\overline{\delta}} (\mathcal{K}_N)_n
\end{array}
$$

which is true by definition of δ. □

Theorem 29 $\mathrm{DIAGR}(\mathcal{C})$ *is finitely cocomplete, with chosen equiv-colimits.*

Proof. We show that the arrow $\overline{\overline{\mathcal{K}}} : \overline{\overline{\Delta}} \to I(\overline{\delta})$ is an equiv-colimiting cone from $\overline{\overline{\Delta}}$. Let $\overline{\overline{\mathcal{Q}}} : \overline{\overline{\Delta}} \to I(\overline{\alpha})$ be another cone from $\overline{\overline{\Delta}}$. We define an arrow $\overline{\mathcal{UP}} : \overline{\delta} \to \overline{\alpha}$ as follows.

– $\mathcal{UP}^\Phi : \delta^\Phi \rightsquigarrow \alpha^\Phi$ is a generalized graph morphism.

  - $\forall (N,n) \in \text{Vertices}(\delta^\Phi),\ \mathcal{UP}^\Phi(N,n) = \mathcal{Q}_N^\Phi(n)$.
  - $\forall (N,a) : (N,n) \to (N,n') \in \text{Edges}(\delta^\Phi)$,
  $$\mathcal{UP}^\Phi(N,a) = \mathcal{Q}_N^\Phi(a) : \mathcal{Q}_N^\Phi(n) \to \mathcal{Q}_N^\Phi(n').$$
  - $\forall (A,n) : (N,n) \to (N', \Delta(A)^\Phi(n)) \in \text{Edges}(\delta^\Phi),\ \mathcal{UP}^\Phi(A,n) = Z$, where $Z : \mathcal{Q}_N^\Phi(n) \rightsquigarrow \mathcal{Q}_{N'}^\Phi(\Delta(A)(n))$ is the zigzag such that
  $$(\mathcal{Q}_{N'})_{\Delta(A)^\Phi(n)} \circ \Delta(A)_n \sim_{\overline{\alpha}} (\mathcal{Q}_N)_n \ \ [Z].$$

– $\mathcal{UP} : \delta \overset{\cdot}{\rightsquigarrow} \alpha \circ \mathcal{P}(\mathcal{UP}^\Phi)$ is the generalized natural transformation defined by

$$\forall (N,n) \in \text{Vertices}(\delta^\Phi),\ \mathcal{UP}_{(N,n)} = (\mathcal{Q}_N)_n \circ (\mathcal{K}_N)_n^{-1}.$$

We must show that $I(\overline{\mathcal{UP}}) \circ \overline{\overline{\mathcal{K}}} \approxeq \overline{\overline{\mathcal{Q}}}$.

$$\begin{aligned} & I(\overline{\mathcal{UP}}) \circ \overline{\overline{\mathcal{K}}} \approxeq \overline{\overline{\mathcal{Q}}} \\ \Leftrightarrow\ & \forall N \in \text{Vertices}(\Delta^\Phi),\ \overline{\mathcal{UP}} \circ \overline{\mathcal{K}_N} \approx \overline{\mathcal{Q}_N} \\ \Leftrightarrow\ & \forall N,\ \forall n \in \text{Vertices}(\Delta(N)^\Phi),\ \mathcal{UP}_{(N,n)} \circ (\mathcal{K}_N)_n \sim_{\overline{\alpha}} (\mathcal{Q}_N)_n \end{aligned}$$

This last statement is true, by definition of $\mathcal{UP}_{(N,n)}$.

It remains to show that $\overline{\mathcal{UP}}$ is unique up to equivalence. Let $\overline{\tau} : \overline{\delta} \to \overline{\alpha}$ such that $I(\overline{\tau}) \circ \overline{\overline{\mathcal{K}}} \approxeq \overline{\overline{\mathcal{Q}}}$. Then,

$$\begin{aligned} & I(\overline{\mathcal{UP}}) \circ \overline{\overline{\mathcal{K}}} \approxeq I(\overline{\tau}) \circ \overline{\overline{\mathcal{K}}} \\ \Rightarrow\ & \forall N \in \text{Vertices}(\Delta^\Phi),\ \overline{\mathcal{UP}} \circ \overline{\mathcal{K}_N} \approx \overline{\tau} \circ \overline{\mathcal{K}_N} \\ \Rightarrow\ & \forall N,\ \forall n \in \text{Vertices}(\Delta(N)^\Phi),\ \mathcal{UP}_{(N,n)} \circ (\mathcal{K}_N)_n \sim_{\overline{\alpha}} \tau_{(N,n)} \circ (\mathcal{K}_N)_n \\ \Rightarrow\ & \forall N,\ \forall n \in \text{Vertices}(\Delta(N)^\Phi),\ \mathcal{UP}_{(N,n)} \sim_{\overline{\alpha}} \tau_{(N,n)} \\ \Rightarrow\ & \overline{\mathcal{UP}} \approx \overline{\tau} \end{aligned}$$

At last, the cone $\overline{\overline{\mathcal{K}}} : \overline{\overline{\Delta}} \to I(\overline{\delta})$ and the mediating arrows $\overline{\mathcal{UP}} : \overline{\delta} \to \overline{\alpha}$ define *chosen* equiv-colimits. □

Application to pushouts

Let $\overline{\alpha}$, $\overline{\beta}$, $\overline{\gamma}$ be three objects of DIAGR($\mathcal{C}$) and $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$, $\overline{\tau} : \overline{\alpha} \to \overline{\gamma}$ be two arrows of DIAGR($\mathcal{C}$). We consider the pushout diagram $\overline{\overline{\Delta}} = \mathcal{PushDiagr}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau})$ in DIAGR^2($\mathcal{C}$). A chosen equiv-colimit of $\overline{\overline{\Delta}}$ is given by the diagram

$$\mathcal{PUSH}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau}) \ = \ \text{Apl}\,\overline{\overline{\Delta}}.$$

The arrows which make up the equiv-colimiting cone from $\overline{\overline{\Delta}}$ are noted

$$\begin{aligned} & \overline{\mathcal{K}_1}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau}) : \overline{\beta} \to \mathcal{PUSH}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau}) \\ & \overline{\mathcal{K}_2}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau}) : \overline{\gamma} \to \mathcal{PUSH}(\overline{\alpha},\overline{\beta},\overline{\gamma},\overline{\sigma},\overline{\tau}). \end{aligned}$$

Given two arrows $\overline{\sigma}' : \overline{\beta} \to \overline{\delta'}$ and $\overline{\tau}' : \overline{\gamma} \to \overline{\delta'}$ such that $\overline{\sigma}' \circ \overline{\sigma} \approx \overline{\tau}' \circ \overline{\tau}$, the mediating arrow from $\mathcal{PUSH}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau})$ to $\overline{\delta'}$ is noted

$$\overline{\mathcal{UP}}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\delta'}, \overline{\sigma}, \overline{\tau}, \overline{\sigma}', \overline{\tau}') : \mathcal{PUSH}(\overline{\alpha}, \overline{\beta}, \overline{\gamma}, \overline{\sigma}, \overline{\tau}) \to \overline{\delta'}.$$

Lemma 30 *The equiv-functor* $\mathrm{Colim} : \mathrm{DIAGR}^2(\mathcal{C}) \to \mathrm{DIAGR}(\mathcal{C})$ *is such that*

$$\mathrm{Colim} \circ \mathrm{DIAGR}(I) \cong Id_{\mathrm{DIAGR}(\mathcal{C})}.$$

Theorem 31 (Completion) $\mathrm{DIAGR}(\mathcal{C})$ *is a* completion *of $\mathcal{C}$ by finite equiv-colimits. In other words, let $\mathcal{C}'$ be a finitely cocomplete equiv-category. Let $F : \mathcal{C} \to \mathcal{C}'$ be an equiv-functor. Then there exists an equiv-functor $G : \mathrm{DIAGR}(\mathcal{C}) \to \mathcal{C}'$, unique up to natural isomorphism, which preserves equiv-colimits and such that $G \circ I \cong F$.*

Proof. Let $G = \mathrm{Colim} \circ \mathrm{DIAGR}(F)$. We have

$$\begin{array}{lll} \mathrm{Colim} \circ I \cong Id_{\mathcal{C}} & & \text{(proposition 20.4)} \\ \quad \Rightarrow & \mathrm{Colim} \circ I \circ F \cong F & \\ \quad \Rightarrow & \mathrm{Colim} \circ \mathrm{DIAGR}(F) \circ I \cong F & \text{(lemma 23)} \\ \quad \Rightarrow & G \circ I \cong F & \text{(definition of } G\text{)} \end{array}$$

We now show that G preserves equiv-colimits, i.e. that given a diagram $\overline{\overline{\Delta}}$ with an equiv-colimiting cone $\overline{\overline{\mathcal{K}}} : \overline{\overline{\Delta}} \to I(\overline{\delta})$, then $G\overline{\overline{\mathcal{K}}} : G \circ \overline{\overline{\Delta}} \to I(G(\overline{\delta}))$ is an equiv-colimiting cone from $G \circ \overline{\overline{\Delta}}$. For all $(N, n) \in \mathrm{Vertices}(\delta^{\Phi})$, let

$$\xi_{(N,n)} = (\eta_{F \circ \overline{\Delta(N)}})_n \circ F((\mathcal{K}_N)_n^{-1}).$$

This defines an arrow $\overline{\xi} : F \circ \overline{\delta} \to G \circ \overline{\overline{\Delta}}$ such that $G\overline{\overline{\mathcal{K}}} \circ \overline{\xi} \approx \overline{\eta}_{F \circ \overline{\delta}}$.
Given a cone $\overline{\mu} : G \circ \overline{\overline{\Delta}} \to I(A)$, we must show that there exists an arrow $\psi : G(\overline{\delta}) \to A$, unique up to equivalence, such that $I(\psi) \circ G\overline{\overline{\mathcal{K}}} \approx \overline{\mu}$.

Existence. As there is a cone $\overline{\mu} \circ \overline{\xi} : F \circ \overline{\delta} \to I(A)$, there exists an arrow $\psi : G(\overline{\delta}) \to A$, unique up to equivalence, such that $I(\psi) \circ \overline{\eta}_{F \circ \overline{\delta}} \approx \overline{\mu} \circ \overline{\xi}$.

$$\begin{array}{ll} I(\psi) \circ \overline{\eta}_{F \circ \overline{\delta}} \approx \overline{\mu} \circ \overline{\xi} & \\ \quad \Rightarrow & \forall (N, n),\ \psi \circ (\eta_{F \circ \overline{\delta}})_{(N,n)} \sim \mu_N \circ \xi_{(N,n)} \\ \quad \Rightarrow & \forall (N, n),\ \psi \circ (\eta_{F \circ \overline{\delta}})_{(N,n)} \sim \mu_N \circ (\eta_{F \circ \overline{\Delta(N)}})_n \circ F((\mathcal{K}_N)_n)^{-1} \\ \quad \Rightarrow & \forall (N, n),\ \psi \circ (\eta_{F \circ \overline{\delta}})_{(N,n)} \circ F((\mathcal{K}_N)_n) \sim \mu_N \circ (\eta_{F \circ \overline{\Delta(N)}})_n \\ \quad \Rightarrow & \forall (N, n),\ \psi \circ G(\overline{\mathcal{K}_N}) \circ (\eta_{F \circ \overline{\Delta(N)}})_n \sim \mu_N \circ (\eta_{F \circ \overline{\Delta(N)}})_n \\ \quad \Rightarrow & \forall N,\ I(\psi \circ G(\overline{\mathcal{K}_N})) \circ \overline{\eta}_{F \circ \overline{\Delta(N)}} \approx I(\mu_N) \circ \overline{\eta}_{F \circ \overline{\Delta(N)}} \\ \quad \Rightarrow & \forall N,\ \psi \circ G(\overline{\mathcal{K}_N}) \sim \mu_N \\ \quad \Rightarrow & I(\psi) \circ G\overline{\overline{\mathcal{K}}} \approx \overline{\mu} \end{array}$$

Unicity. Let $\psi' : G(\overline{\delta}) \to A$ such that $I(\psi') \circ G\overline{\overline{\mathcal{K}}} \approx \overline{\mu}$. Then,

$$\begin{array}{l} I(\psi') \circ \overline{\eta}_{F \circ \overline{\delta}} \ \approx\ I(\psi') \circ G\overline{\overline{\mathcal{K}}} \circ \overline{\xi} \ \approx\ \overline{\mu} \circ \overline{\xi} \\ \quad \Rightarrow \quad \psi \sim \psi' \end{array}$$

At last, we must show that G is unique up to natural isomorphism. Let $H : \mathrm{DIAGR}(\mathcal{C}) \to \mathcal{C}'$ be an equiv-functor which preserves equiv-colimits and such that $H \circ I \cong F$.

$$\begin{array}{lll} G \circ I \cong H \circ I & & \\ \quad \Rightarrow & \mathrm{Colim} \circ \mathrm{DIAGR}(G) \circ \mathrm{DIAGR}(I) \cong \mathrm{Colim} \circ \mathrm{DIAGR}(H) \circ \mathrm{DIAGR}(I) & \\ \quad \Rightarrow & G \circ \mathrm{Colim} \circ \mathrm{DIAGR}(I) \cong H \circ \mathrm{Colim} \circ \mathrm{DIAGR}(I) & \text{(lemma 25)} \\ \quad \Rightarrow & G \cong H & \text{(lemma 30)} \end{array}$$

□

3.5 From equiv-categories to categories

The definitions of equiv-category, equiv-functor and equiv-colimit lead to the usual definitions of category, functor and colimit.

Category If $\mathcal{C} = (\mathrm{Obj}(\mathcal{C}), \mathrm{Arr}(\mathcal{C}), \sim)$ is an equiv-category, then there is a *category* $\mathcal{C}/\!\sim$ whose class of objects is $\mathrm{Obj}(\mathcal{C})$ and whose set of arrows from an object A to an object B is the quotient set $\mathrm{Arr}(\mathcal{C})(A, B)/\!\sim$.

There is a projection equiv-functor $P_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}/\!\sim$ which is the identity on objects and which assigns to each arrow $f \in \mathrm{Arr}(\mathcal{C})(A, B)$ its equivalence class $[f] : A \to B$ in $\mathcal{C}/\!\sim$.

Reciprocally, we can consider any category as an equiv-category by taking the equality relation on arrows as congruence relation.

Functor Any equiv-functor between two equiv-categories gives rise to a *functor* between the corresponding categories. Let $F : \mathcal{C} \to \mathcal{C}'$ be an equiv-functor. Then there is a unique *functor* $F/\!\sim : \mathcal{C}/\!\sim \to \mathcal{C}'/\!\sim$, which is equal to F on objects and such that

$$\forall f \in \mathrm{Arr}(\mathcal{C})(A, B),\ (F/\!\sim)([f]) = [F(f)].$$

An equiv-functor $F : \mathcal{C} \to \mathcal{C}'$ is full (respectively faithful) if and only if the functor $F/\!\sim$ is full (respectively faithful).

Natural transformation between functors Let $F, G : \mathcal{C} \to \mathcal{C}'$ be two equiv-functors between the equiv-categories $\mathcal{C}$ and $\mathcal{C}'$. Let $\sigma : F \dot{\to} G$ be a natural transformation. Then, there is a natural transformation $[\sigma] : F/\!\sim \dot{\to} G/\!\sim$ defined by $[\sigma]_A = [\sigma_A]$.

Two equiv-functors F and G are naturally isomorphic if and only if $F/\!\sim$ and $G/\!\sim$ are naturally isomorphic.

Colimit Let $\mathcal{C}$ be an equiv-category. An arrow $\overline{\lambda} : \overline{\alpha} \to I(C)$ of $\mathrm{DIAGR}(\mathcal{C})$ is an equiv-colimiting cone from a diagram $\overline{\alpha}$ if and only if the arrow $P_{\mathcal{C}}\overline{\lambda} : P_{\mathcal{C}} \circ \overline{\alpha} \to I(C)$ of $\mathrm{DIAGR}(\mathcal{C}/\!\sim)$ is a colimiting cone from $P_{\mathcal{C}} \circ \overline{\alpha}$.

Therefore, an equiv-category $\mathcal{C}$ is finitely cocomplete if and only if the category $\mathcal{C}/\!\sim$ if finitely cocomplete.

As a category is finitely cocomplete if and only if it has an initial object and pushouts, an equiv-category $\mathcal{C}$ is finitely cocomplete if and only if $\mathcal{C}$ has an equiv-initial object and equiv-pushouts.

Remark 32 (Chosen colimits versus chosen equiv-colimits)
In an equiv-category, a choice of equiv-colimit for a diagram $\overline{\alpha}$ consists of

1. a choice of an equiv-colimiting cone $\overline{\lambda} : \overline{\alpha} \to I(C)$;

2. for any other cone $\overline{\mu} : \overline{\alpha} \to I(D)$, a choice of arrow $\psi : C \to D$ such that $I(\psi) \circ \overline{\lambda} \approx \overline{\mu}$ (this arrow is only unique *up to equivalence*).

In a category though, a choice of colimit for a diagram $\overline{\alpha}$ just consists of a choice of a colimiting cone $\overline{\lambda} : \overline{\alpha} \to I(C)$. Then for any other cone $\overline{\mu} : \overline{\alpha} \to I(D)$, there exists a *unique* $\psi : C \to D$ such that $I(\psi) \circ \overline{\lambda} = \overline{\mu}$.

We must note that a choice of equiv-colimits in an equiv-category $\mathcal{C}$ does not induce a choice of colimits in the category $\mathcal{C}/\!\sim$. Indeed, given a diagram $\overline{\alpha}$ over $\mathcal{C}/\!\sim$, there may exist several diagrams $\overline{\beta}$ over $\mathcal{C}$ such that $P_{\mathcal{C}} \circ \overline{\beta} = \overline{\alpha}$, which induce *several different possible choices* for the colimit of $\overline{\alpha}$.

3.6 Category of diagrams Diagr($\mathcal{C}$)

The category of diagrams Diagr($\mathcal{C}$) is the quotient category corresponding to the equiv-category DIAGR($\mathcal{C}$)

$$\mathrm{Diagr}(\mathcal{C}) = \mathrm{DIAGR}(\mathcal{C})/\!\approx.$$

This category is a completion of $\mathcal{C}$ by finite colimits. Diagr($\mathcal{C}$) is thus finitely cocomplete, but has no chosen colimits (cf. remark 32).

The category Diagr($\mathcal{C}$) is well known in category theory. Its objects are diagrams. The set of arrows from a diagram $\overline{\alpha}$ to a diagram $\overline{\beta}$ may be defined "concretely" with limits and colimits of hom-functors as

$$\mathrm{Hom}(\overline{\alpha}, \overline{\beta}) = \operatorname*{Lim}_{x \in \alpha^{\Phi}} \ \operatorname*{Colim}_{y \in \beta^{\Phi}} \mathrm{Hom}(\alpha(x), \beta(y)).$$

Our definition is very similar to the "abstract" definition proposed in [35] (see also [2]), except that in [35], a diagram morphism $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ consists of a set of

$$\{\sigma_n,\ n \in \mathrm{Vertices}(\alpha^{\Phi})\}$$

where σ_n is an *equivalence class* of arrows modulo $\sim_{\overline{\beta}}$ from $\alpha(n)$ to $\beta(n')$.

In this section, we actually presented the category Diagr($\mathcal{C}$) as a quotient of the "syntactic" equiv-category DIAGR($\mathcal{C}$) by $\approx$. DIAGR($\mathcal{C}$) is "syntactic" in the sense that we have an effective representation of its arrows. The advantage of our (long) definition is be able to manipulate arrows of Diagr($\mathcal{C}$) by representatives taken in DIAGR($\mathcal{C}$). Therefore, all computations (like those described in section 5) take place in the equiv-category DIAGR($\mathcal{C}$), while the results may be interpreted in the category Diagr($\mathcal{C}$).

4 The Term Language TERM($\mathcal{C}_0$)

The aim of this section is to define a *term language* for describing colimit constructions. We build an equiv-category TERM($\mathcal{C}_0$) whose objects represent colimit constructions, and arrows colimiting arrows between colimit constructions.

We wish to be able to represent any colimit constructions of base specifications related by base specification morphisms. As a category is finitely cocomplete if and only if it has an initial object and pushouts, we choose to have a representation for these two constructions.

Therefore, we define a term **Ø** to represent the initial object, and for all objects A, B, C and arrows $f : A \to B$, $g : A \to C$, we define a term $\mathsf{push}(A, B, C, f, g)$ to represent the pushout of the diagram $\mathcal{PushDiagr}(A, B, C, f, g)$. We will also need terms to denote specification morphisms.

4.1 *Problem of circularity*

Let $\mathcal{C}_0$ be a small category (that we will consider as an equiv-category). The aim is to define an equiv-category which contains $\mathcal{C}_0$, an equiv-initial object and equiv-pushouts.

Equiv-initial object

For the equiv-initial object, we just have to introduce a new object $\emptyset$ and for every object A an arrow $\mathsf{j}(A) : \emptyset \to A$. Moreover, we also need to introduce for all $f, g : \emptyset \to A$ the relation $f \sim g$.

Equiv-pushouts

Given three objects A, B, C and two arrows $f : A \to B$, $g : A \to C$, we need to introduce a new object

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

two arrows

$$\begin{array}{l} \mathsf{k}_1(A, B, C, f, g) : B \to \mathsf{push}(A, B, C, f, g) \\ \mathsf{k}_2(A, B, C, f, g) : C \to \mathsf{push}(A, B, C, f, g), \end{array}$$

and the relation

$$\mathsf{k}_1(A, B, C, f, g) \circ f \sim \mathsf{k}_2(A, B, C, f, g) \circ g.$$

Moreover, given two arrows $f' : B \to D$ and $g' : C \to D$ such that $f' \circ f \sim g' \circ g$, we need to introduce an arrow

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

and two relations

$$\begin{array}{l} \mathsf{up}(A, B, C, D, f, g, f', g') \circ \mathsf{k}_1(A, B, C, f, g) \sim f' \\ \mathsf{up}(A, B, C, D, f, g, f', g') \circ \mathsf{k}_2(A, B, C, f, g) \sim g'. \end{array}$$

Introducing the arrow $\mathsf{up}(A, B, C, D, f, g, f', g')$ raises a problem. Until now, we have first defined terms and then relations on these terms. Here, we need to introduce a new term *only if a relation is satisfied*. There is therefore a circularity between the definition of terms and the definition of relations.

Generalized algebraic theories, which are a generalization of multi-sorted algebras, have been proposed by Cartmell to specify *dependent types*, i.e. types parameterized by terms [9]. For instance, the "type" $\mathrm{Arr}(A, B)$ depends on both terms A and B. J.-C. Reynaud proposes to use Cartmell's dependent types to specify colimit constructions [30, 31, 32]. The syntax presented here is widely inspired by his work. However, Cartmell's dependent types cannot specify terms which are conditioned by a relation between two other terms. J.-C. Reynaud gets round the difficulty by constructing a concrete, i.e. semantic, finitely cocomplete category [32]. T. Streicher and M. Wirsing, who also advocate the use of dependent types to describe colimit constructions [33] do not make it clear how they solve this circularity problem.

Besides, H. Ehrig *et al.* [15] also propose a syntax to describe colimit constructions. But as their syntax has no representation for all up arrows, it is not powerful enough to describe all colimit constructions.

One solution is to specify a finitely cocomplete equiv-category without up arrows, by replacing them by other arrows whose existence is not conditioned by any relation. This approach has been proposed by F. Cury [10], who introduces two arrows called p and d, which do not depend on any relation, and allow to reconstruct *a posteriori* any up arrow.

Here, we wish to stay close to the classic definition of pushouts and therefore to keep the up arrows. For this reason, we propose a stratified construction of the equiv-category $\mathrm{TERM}(\mathcal{C}_0)$, by defining a sequence of equiv-categories $(\mathcal{C}_i)_{i\geq 0}$ *such that the introduction of an* up *arrow in an equiv-category* $\mathcal{C}_i$ *only depends on a relation in* $\mathcal{C}_{i-1}$. The equiv-category of terms is then the union of all equiv-categories $\mathcal{C}_i$.

4.2 Equiv-categories $\mathcal{C}_i$

The sequence of equiv-categories $\mathcal{C}_i = (\mathrm{Obj}(\mathcal{C}_i), \mathrm{Arr}(\mathcal{C}_i), \sim_i)$ is defined by induction over $i \geq 1$ by the following rules.

Rules defining the set $\mathrm{Obj}(\mathcal{C}_i)$

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

$$\frac{}{\emptyset \in \mathrm{Obj}(\mathcal{C}_i)} \tag{2}$$

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

Rules defining the family of sets $\mathrm{Arr}(\mathcal{C}_i)$

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

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

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_i)}{\mathsf{id}(A) \in \mathrm{Arr}(\mathcal{C}_i)(A, A)} \tag{6}$$

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_i)}{\mathsf{j}(A) \in \mathrm{Arr}(\mathcal{C}_i)(\emptyset, A)} \tag{7}$$

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

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

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

Rules defining the family of relations $\sim_i$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_0) \;\; ; \;\; f, g \in \mathrm{Arr}(\mathcal{C}_0)(A, B) \;\; ; \;\; f \sim_0 g}{f \sim_i g} \tag{11}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)}{f \sim_i f} \tag{12}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f, g \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \;\; ; \;\; f \sim_i g}{g \sim_i f} \tag{13}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f, g, h \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \;\; ; \;\; f \sim_i g \;\; ; \;\; g \sim_i h}{f \sim_i h} \tag{14}$$

$$\frac{\begin{array}{c} A, B, C \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f, f' \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \\ g, g' \in \mathrm{Arr}(\mathcal{C}_i)(B, C) \;\; ; \;\; f \sim_i f' \;\; ; \;\; g \sim_i g' \end{array}}{g \circ f \sim_i g' \circ f'} \tag{15}$$

$$\frac{\begin{array}{c} A, B, C, D \in \mathrm{Obj}(\mathcal{C}_i) \\ f \in \mathrm{Arr}(\mathcal{C}_i)(A, B) \;\; ; \;\; g \in \mathrm{Arr}(\mathcal{C}_i)(B, C) \;\; ; \;\; h \in \mathrm{Arr}(\mathcal{C}_i)(C, D) \end{array}}{(h \circ g) \circ f \sim_i h \circ (g \circ f)} \tag{16}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)}{f \circ \mathsf{id}(A) \sim_i f} \tag{17}$$

$$\frac{A, B \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_i)(A, B)}{\mathsf{id}(B) \circ f \sim_i f} \tag{18}$$

$$\frac{A \in \mathrm{Obj}(\mathcal{C}_i) \;\; ; \;\; f, g \in \mathrm{Arr}(\mathcal{C}_i)(\varnothing, A)}{f \sim_i g} \tag{19}$$

$$\frac{A, B, C \in \mathrm{Obj}(\mathcal{C}_{i-1}) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, B) \;\; ; \;\; g \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, C)}{\mathsf{k}_1(A, B, C, f, g) \circ f \sim_i \mathsf{k}_2(A, B, C, f, g) \circ g} \tag{20}$$

$$\frac{\begin{array}{c} A, B, C, D \in \mathrm{Obj}(\mathcal{C}_{i-1}) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, B) \;\; ; \;\; g \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, C) \\ f' \in \mathrm{Arr}(\mathcal{C}_{i-1})(B, D) \;\; ; \;\; g' \in \mathrm{Arr}(\mathcal{C}_{i-1})(C, D) \;\; ; \;\; f' \circ f \sim_{i-1} g' \circ g \end{array}}{\mathsf{up}(A, B, C, D, f, g, f', g') \circ \mathsf{k}_1(A, B, C, f, g) \sim_i f'} \tag{21}$$

$$\frac{\begin{array}{c} A, B, C, D \in \mathrm{Obj}(\mathcal{C}_{i-1}) \;\; ; \;\; f \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, B) \;\; ; \;\; g \in \mathrm{Arr}(\mathcal{C}_{i-1})(A, C) \\ f' \in \mathrm{Arr}(\mathcal{C}_{i-1})(B, D) \;\; ; \;\; g' \in \mathrm{Arr}(\mathcal{C}_{i-1})(C, D) \;\; ; \;\; f' \circ f \sim_{i-1} g' \circ g \end{array}}{\mathsf{up}(A, B, C, D, f, g, f', g') \circ \mathsf{k}_2(A, B, C, f, g) \sim_i g'} \tag{22}$$

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

Lemma 33 *For all $i \geq 0$,*

1. $\mathrm{Obj}(\mathcal{C}_i) \subseteq \mathrm{Obj}(\mathcal{C}_{i+1})$;

2. $\mathrm{Arr}(\mathcal{C}_i)(A, B) \subseteq \mathrm{Arr}(\mathcal{C}_{i+1})(A, B)$;

3. $\forall f, g \in \mathrm{Arr}(\mathcal{C}_i)(A, B), \; f \sim_i g \;\Rightarrow\; f \sim_{i+1} g$.

Proof. By induction on i. This result is obvious for $i = 0$. For the inductive step, we prove the three points in parallel, by structural induction on the definition of $\mathrm{Obj}(\mathcal{C}_i)$, $\mathrm{Arr}(\mathcal{C}_i)$ and $\sim_i$. □

Lemma 34 *For all $i \geq 0$,*

1. *$\mathcal{C}_i$ is an equiv-category ;*
2. *if $i \geq 1$, then $\mathbf{\emptyset}$ is equiv-initial in $\mathcal{C}_i$.*

Proof. Obvious from the rules defining $\mathrm{Obj}(\mathcal{C}_i)$, $\mathrm{Arr}(\mathcal{C}_i)$, and $\sim_i$. □

Remark 35 The statement $\mathit{push}(A, B, C, f, g) \in \mathrm{Obj}(\mathcal{C}_i)$ does not mean that the object $\mathit{push}(A, B, C, f, g)$ is an equiv-pushout in $\mathcal{C}_i$. Indeed we have delayed the introduction of some *up* arrows in order to avoid circularity in our definition.

For all $i \geq 0$, the equiv-category $\mathcal{C}_i$ is a *conservative extension* of $\mathcal{C}_0$, which means that we do not introduce in $\mathcal{C}_i$ new arrows between objects of $\mathcal{C}_0$, and that we do not introduce new relations between arrows of $\mathcal{C}_0$.

For all $i \geq 0$, let J_i be the inclusion equiv-functor of $\mathcal{C}_0$ into $\mathcal{C}_i$.

Theorem 36 ($\mathcal{C}_i$ is a conservative extension of $\mathcal{C}_0$)
For all $i \geq 0$, the equiv-functor $J_i : \mathcal{C}_0 \to \mathcal{C}_i$ is full and faithful.

This theorem is equivalent to the following result:
$\forall i \geq 0,\ \forall A, B \in \mathrm{Obj}(\mathcal{C}_0)$,

1. (J_i is full) $\forall h \in \mathrm{Arr}(\mathcal{C}_i)(A, B),\ \exists h' \in \mathrm{Arr}(\mathcal{C}_0)(A, B)\ ;\ h \sim_i h'$
2. (J_i is faithful) $\forall h', h'' \in \mathrm{Arr}(\mathcal{C}_0)(A, B),\ h' \sim_i h'' \ \Rightarrow\ h' \sim_0 h''$

The proof, which consists of tedious inductions, is too long to be written here and is developed in [28].

4.3 Equiv-category of terms TERM($\mathcal{C}_0$)

Definition 37 We define the equiv-category of terms

$$\mathrm{TERM}(\mathcal{C}_0) = (\mathrm{Obj}(\mathrm{TERM}(\mathcal{C}_0)), \mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0)), \sim)$$

as the union of all equiv-categories $\mathcal{C}_i$.

- The set of objects of TERM($\mathcal{C}_0$) is
$$\mathrm{Obj}(\mathrm{TERM}(\mathcal{C}_0)) = \bigcup_{i=0}^{\infty} \mathrm{Obj}(\mathcal{C}_i).$$
- Let $A, B \in \mathrm{Obj}(\mathrm{TERM}(\mathcal{C}_0))$. There exists $k \geq 0$ such that $A, B \in \mathrm{Obj}(\mathcal{C}_k)$. The set of arrows from A to B is
$$\mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0))(A, B) = \bigcup_{i=k}^{\infty} \mathrm{Arr}(\mathcal{C}_i)(A, B).$$

- Let $f, g \in \mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0))(A, B)$. There exists $k \geq 0$ such that $f, g \in \mathrm{Arr}(\mathcal{C}_k)(A, B)$. By definition, $f \sim g$ if and only if $f \sim_k g$.

From lemma 33, $\mathrm{Obj}(\mathrm{TERM}(\mathcal{C}_0))$, $\mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0))$ and $\sim$ are well defined, and from lemma 34.1, $\mathrm{TERM}(\mathcal{C}_0)$ is an equiv-category.

Theorem 38 $\mathrm{TERM}(\mathcal{C}_0)$ *is a finitely cocomplete equiv-category.*

Proof. The object $\boldsymbol{\emptyset}$ is equiv-initial in $\mathrm{TERM}(\mathcal{C}_0)$. $\forall A, B, C \in \mathrm{Obj}(\mathrm{TERM}(\mathcal{C}_0))$, $f \in \mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0))(A, B)$, $g \in \mathrm{Arr}(\mathrm{TERM}(\mathcal{C}_0))(A, C)$, the triple

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

is an equiv-pushout. Therefore, $\mathrm{TERM}(\mathcal{C}_0)$ has all finite equiv-colimits. □

Let $J : \mathcal{C}_0 \to \mathrm{TERM}(\mathcal{C}_0)$ be the inclusion equiv-functor of $\mathcal{C}_0$ into $\mathrm{TERM}(\mathcal{C}_0)$.

Theorem 39 ($\mathrm{TERM}(\mathcal{C}_0)$ is a conservative extension of $\mathcal{C}_0$)
The equiv-functor $J : \mathcal{C}_0 \to \mathrm{TERM}(\mathcal{C}_0)$ is full and faithful.

Proof. This is a consequence of theorem 36. □

Theorem 40 $\mathrm{TERM}(\mathcal{C}_0)$ *is the equiv-category freely generated over $\mathcal{C}_0$ by a chosen equiv-initial object and chosen equiv-pushouts. In other words, let $F : \mathcal{C}_0 \to \mathcal{E}$ be an equiv-functor, $\mathcal{E}$ be an equiv-category with a chosen equiv-initial object and chosen equiv-pushouts. Then there exists a unique equiv-functor*

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

which preserves the chosen equiv-initial object and chosen equiv-pushouts and such that

$$G \circ J = F.$$

Proof sketch. We construct the equiv-functor $G : \mathrm{TERM}(\mathcal{C}_0) \to \mathcal{E}$ by induction on the structure of objects and arrows of $\mathrm{TERM}(\mathcal{C}_0)$, such that

1. $\forall A \in \mathrm{Obj}(\mathcal{C}_0),\ G(A) = F(A)$
2. $G(\boldsymbol{\emptyset}) = \emptyset^{\mathcal{E}}$
3. $G(\mathsf{push}(A, B, C, f, g)) = push^{\mathcal{E}}(G(A), G(B), G(C), G(f), G(g))$
4. $\forall f \in \mathrm{Arr}(\mathcal{C}_0)(A, B),\ G(f) = F(f)$
5. $G(g \circ f) = G(g) \circ G(f)$
6. $G(\mathsf{id}(A)) = id_{G(A)}$
7. $G(\mathsf{j}(A)) = j^{\mathcal{E}}_A$
8. $G(\mathsf{k}_1(A, B, C, f, g)) = k^{\mathcal{E}}_1(G(A), G(B), G(C), G(f), G(g))$
9. $G(\mathsf{k}_2(A, B, C, f, g)) = k^{\mathcal{E}}_2(G(A), G(B), G(C), G(f), G(g))$

10. $G(\mathsf{up}(A,B,C,D,f,g,f',g'))$
 $= \mathsf{up}^{\mathcal{E}}(G(A),G(B),G(C),G(D),G(f),G(g),G(f'),G(g'))$.

We show that G is compatible with the congruences in $\mathrm{TERM}(\mathcal{C}_0)$ and $\mathcal{E}$ by induction on the length of the proof that $h \sim h'$ in $\mathrm{TERM}(\mathcal{C}_0)$. Then, G is an equiv-functor because of conditions 5 and 6; $G \circ J = F$ because of 1 and 4; G preserves the chosen initial object because of 2 and 7; G preserves chosen equiv-pushouts because of 3, 8, 9 and 10. □

4.4 Category of terms $\mathrm{Term}(\mathcal{C}_0)$

The category of terms $\mathrm{Term}(\mathcal{C}_0)$ is the quotient category

$$\mathrm{Term}(\mathcal{C}_0) = \mathrm{TERM}(\mathcal{C}_0)/\!\sim.$$

Obviously, from theorem 38, $\mathrm{Term}(\mathcal{C}_0)$ is a finitely cocomplete category. However, the category $\mathrm{Term}(\mathcal{C}_0)$ has no *chosen* pushout. Indeed, if $f \sim f'$ and $g \sim g'$ in $\mathrm{TERM}(\mathcal{C}_0)$, then $\mathsf{push}(A,B,C,f,g)$ and $\mathsf{push}(A,B,C,f',g')$ are two *different choices* of pushouts of $f = f'$ and $g = g'$ in $\mathrm{Term}(\mathcal{C}_0)$ (cf. remark 32). Therefore, $\mathrm{Term}(\mathcal{C}_0)$ is not *freely generated by a chosen initial object and chosen pushouts.* However, it is possible to construct a category freely generated by a chosen initial object and chosen pushouts by identifying the multiple choices of pushouts. This construction, inspired by F. Cury's "object rewriting" [10], is described in [28].

Categories freely generated by certain limits or colimits correspond to the *type of a sketch* introduced by C. Ehresmann [14]. Here, the advantage of defining the equiv-category $\mathrm{TERM}(\mathcal{C}_0)$ is to get an effective representation of arrows.

5 From terms to diagrams

In this section, we show how to associate with each specification (represented by an object of $\mathrm{TERM}(\mathcal{C}_0)$) a diagram, and with each specification morphism (represented by an arrow of $\mathrm{TERM}(\mathcal{C}_0)$) a diagram morphism.

5.1 Equiv-functor $\mathcal{D} : \mathrm{TERM}(\mathcal{C}_0) \to \mathrm{DIAGR}(\mathcal{C}_0)$

The equiv-category $\mathrm{DIAGR}(\mathcal{C}_0)$ has a chosen equiv-initial object and chosen equiv-pushouts (theorem 29). Therefore, from theorem 40, there exists a unique equiv-functor

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

such that

1. $\forall A \in \mathrm{Obj}(\mathcal{C}_0),\ \mathcal{D}(A) = I(A)$
2. $\mathcal{D}(\boldsymbol{\emptyset}) = \bigcirc$
3. $\mathcal{D}(\mathsf{push}(A,B,C,f,g))$
 $= \mathcal{PUSH}(\mathcal{D}(A),\mathcal{D}(B),\mathcal{D}(C),\mathcal{D}(f),\mathcal{D}(g))$
 $= \mathrm{Apl}\,\mathcal{PushDiagr}(\mathcal{D}(A),\mathcal{D}(B),\mathcal{D}(C),\mathcal{D}(f),\mathcal{D}(g))$
4. $\forall f \in \mathrm{Arr}(\mathcal{C}_0)(A,B),\ \mathcal{D}(f) = I(f)$

5. $\mathcal{D}(g \circ f) = \mathcal{D}(g) \circ \mathcal{D}(f)$

6. $\mathcal{D}(\mathsf{id}(A)) = \overline{\mathcal{Id}}_{\mathcal{D}(A)}$

7. $\mathcal{D}(\mathsf{j}(A)) = \overline{\mathcal{J}}_{\mathcal{D}(A)}$

8. $\mathcal{D}(\mathsf{k}_1(A,B,C,f,g)) = \overline{\mathcal{K}_1}(\mathcal{D}(A),\mathcal{D}(B),\mathcal{D}(C),\mathcal{D}(f),\mathcal{D}(g))$

9. $\mathcal{D}(\mathsf{k}_2(A,B,C,f,g)) = \overline{\mathcal{K}_2}(\mathcal{D}(A),\mathcal{D}(B),\mathcal{D}(C),\mathcal{D}(f),\mathcal{D}(g))$

10. $\mathcal{D}(\mathsf{up}(A,B,C,D,f,g,f',g'))$
    $= \overline{\mathcal{UP}}(\mathcal{D}(A),\mathcal{D}(B),\mathcal{D}(C),\mathcal{D}(D),\mathcal{D}(f),\mathcal{D}(g),\mathcal{D}(f'),\mathcal{D}(g'))$.

The rules 1 and 4 are equivalent to $\mathcal{D} \circ J = I$.

These rules give us a procedure to compute the diagram associated with a term. At last, there is of course a functor $\mathcal{D}/\sim\ : \mathrm{Term}(\mathcal{C}_0) \to \mathrm{Diagr}(\mathcal{C}_0)$ which corresponds to the equiv-functor $\mathcal{D} : \mathrm{TERM}(\mathcal{C}_0) \to \mathrm{DIAGR}(\mathcal{C}_0)$.

5.2 Example

As an example, we compute the diagram associated with the specification R_3 of rings which was defined as follows (cf. section 2).

$$\begin{aligned}
P &= \mathsf{push}(S, M, G, b \circ s, m \circ b \circ s)\\
B_2 &= \mathsf{push}(\emptyset, B, B, \mathsf{j}(B), \mathsf{j}(B))\\
u_1 &= \mathsf{up}(\emptyset, B, B, P, \mathsf{j}(B), \mathsf{j}(B), \mathsf{k}_1(P) \circ b, \mathsf{k}_2(P) \circ m \circ b) : B_2 \to P\\
u_2 &= \mathsf{up}(\emptyset, B, B, D, \mathsf{j}(B), \mathsf{j}(B), m_*, m_+) : B_2 \to D\\
R_3 &= \mathsf{push}(B_2, P, D, u_1, u_2)
\end{aligned}$$

Let us start by computing the diagram associated with the specification P of pseudo-rings.

$\mathcal{D}(P) = \mathrm{Apl}\,\mathcal{PushDiagr}(\mathcal{D}(S), \mathcal{D}(M), \mathcal{D}(G), \mathcal{D}(b \circ s), \mathcal{D}(m \circ b \circ s))$

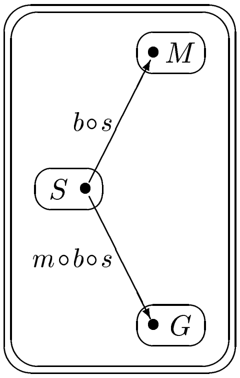
$\mathcal{D}(S) = \boxed{\bullet\ S}$

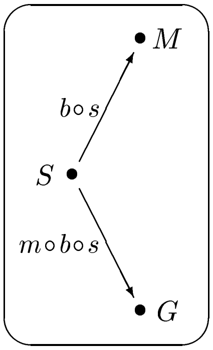
$\mathcal{D}(M) = \boxed{\bullet\ M}$

$\mathcal{D}(G) = \boxed{\bullet\ G}$

$\mathcal{D}(b \circ s) \ = \ \boxed{B\ \bullet} \xrightarrow{b \circ s} \boxed{\bullet\ D}$

$\mathcal{D}(m \circ b \circ s) \ = \ \boxed{B\ \bullet} \xrightarrow{m \circ b \circ s} \boxed{\bullet\ D}$

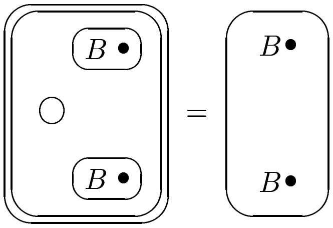
$\mathcal{D}(P) \ = \ \mathrm{Apl}$ 

$=$ 

$= \ \overline{\zeta}$

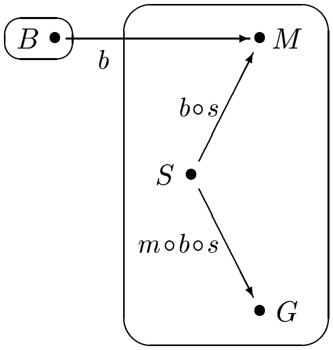
Let us compute the diagram associated with the specification B_2.

$\mathcal{D}(B_2) = \text{Apl}\,\mathcal{P}ush\mathcal{D}iagr(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(j(B)), \mathcal{D}(j(B)))$
$\mathcal{D}(\emptyset) = \bigcirc$
$\mathcal{D}(B) = \boxed{\bullet\ B}$
$\mathcal{D}(j(B)) = \bigcirc \to \boxed{\bullet\ B}$

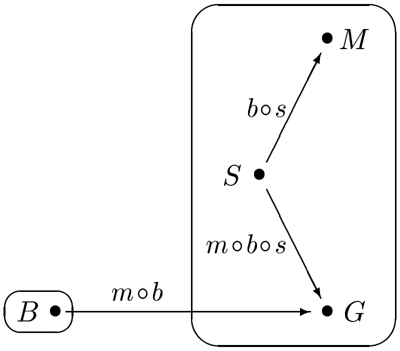
$\mathcal{D}(B_2) \;=\; \text{Apl}$ 

To find the diagram morphism associated with the specification morphism $u_1 : B_2 \to P$, we must first calculate $\mathcal{D}(k_1(P) \circ b)$ and $\mathcal{D}(k_2(P) \circ m \circ b)$.

$\mathcal{D}(k_1(P) \circ b) \;=\; \overline{\mathcal{K}_1}(\mathcal{D}(S), \mathcal{D}(M), \mathcal{D}(G), \mathcal{D}(b \circ s), \mathcal{D}(m \circ b \circ s)) \circ \mathcal{D}(B)$

$=$ 

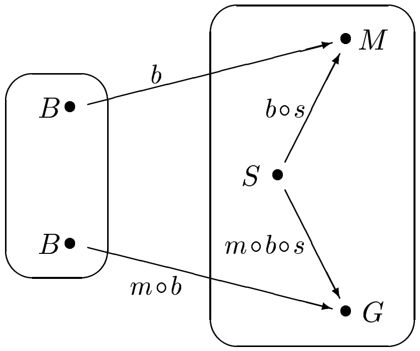
$\mathcal{D}(k_2(P) \circ m \circ b) \;=\; \overline{\mathcal{K}_2}(\mathcal{D}(S), \mathcal{D}(M), \mathcal{D}(G), \mathcal{D}(b \circ s), \mathcal{D}(m \circ b \circ s))$
$\circ\, \mathcal{D}(m) \circ \mathcal{D}(b)$

$=$ 

We can now compute the diagram morphism associated with $u_1 : B_2 \to P$.

$$\begin{aligned}\mathcal{D}(u_1) &= \mathcal{D}(\mathit{up}(\emptyset, B, B, P, j(B), j(B), k_1(P)\circ b, k_2(P)\circ m\circ b))\\ &= \overline{\mathcal{UP}}(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(P),\\ &\qquad \mathcal{D}(j(B)), \mathcal{D}(j(B)), \mathcal{D}(k_1(P)\circ b), \mathcal{D}(k_2(P)\circ m\circ b))\end{aligned}$$

$=$



Let us compute the diagram morphism associated with $u_2 : B_2 \to D$.

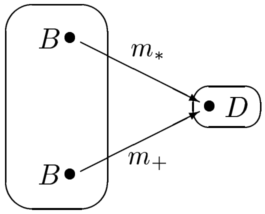
$$\begin{aligned}\mathcal{D}(u_2) &= \mathcal{D}(\mathit{up}(\emptyset, B, B, D, j(B), j(B), m_*, m_+))\\ &= \overline{\mathcal{UP}}(\mathcal{D}(\emptyset), \mathcal{D}(B), \mathcal{D}(B), \mathcal{D}(D), \mathcal{D}(j(B)), \mathcal{D}(j(B)), \mathcal{D}(m_*), \mathcal{D}(m_+))\end{aligned}$$

$\mathcal{D}(D) = \boxed{\bullet\ D}$

$$\mathcal{D}(m_*) = \begin{tikzcd}\boxed{B\ \bullet} \arrow[r, "m_*"] & \boxed{\bullet\ D}\end{tikzcd}$$

$$\mathcal{D}(m_+) = \begin{tikzcd}\boxed{B\ \bullet} \arrow[r, "m_+"] & \boxed{\bullet\ D}\end{tikzcd}$$

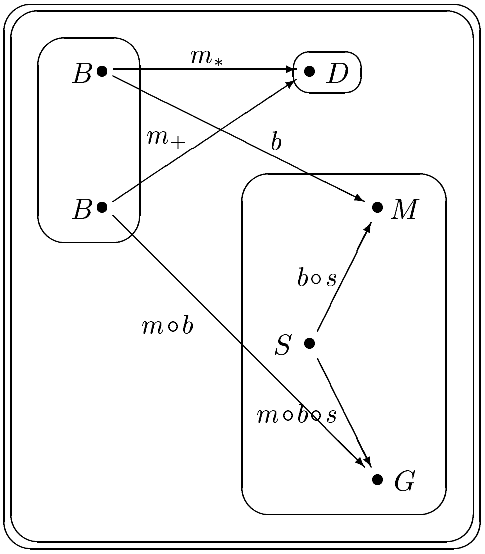
$\mathcal{D}(u_2) =$



At last, we compute the diagram associated with the specification R_3.

$$\begin{aligned}\mathcal{D}(A_3) &= \mathcal{D}(\mathit{push}(B_2, P, D, u_1, u_2)) \\ &= \mathrm{Apl}\,\mathcal{P}\mathit{ush}\mathcal{D}\mathit{iagr}(\mathcal{D}(B_2), \mathcal{D}(P), \mathcal{D}(D), \mathcal{D}(u_1), \mathcal{D}(u_2))\end{aligned}$$

$= \mathrm{Apl}$



$$= \begin{tikzcd}
& & M \\
& B \arrow[ur, "b"'] \arrow[dr, "m_*"] & \\
S \arrow[uurr, "b \circ s"] \arrow[ddrr, "m \circ b \circ s"'] & & D \\
& B \arrow[ur, "m_+"'] \arrow[dr, "m \circ b"] & \\
& & G
\end{tikzcd}$$

$\overline{\delta_3}$

Therefore, the diagram associated with the specification R_3 is $\overline{\delta_3}$ (cf. section 2).

If we compute the diagram associated with R_4, we find a diagram $\overline{\delta'_4}$, which is isomorphic to $\overline{\delta_4}$.

$$\mathcal{D}(A_4) = \begin{tikzcd}
& M & \\
B \arrow[ur, "b"'] \arrow[dr, "id_B"] & & \\
& B & B \arrow[l, "id_B"'] \arrow[d, "m_*"] \\
S \arrow[ur, "s"'] \arrow[dr, "s"] & & D \\
& B & B \arrow[l, "id_B"] \arrow[u, "m_+"'] \\
B \arrow[ur, "id_B"'] \arrow[dr, "m \circ b"] & & \\
& G &
\end{tikzcd}
\cong
\begin{tikzcd}
& & M \\
& B \arrow[ur, "b"'] \arrow[dr, "m_*"] & \\
S \arrow[ur, "s"] \arrow[dr, "s"'] & & D \\
& B \arrow[ur, "m_+"'] \arrow[dr, "m \circ b"] & \\
& & G
\end{tikzcd}$$

$\overline{\delta_4}$

$\overline{\delta'_4}$

5.3 Equivalence of terms and diagrams

In this paragraph, we show that the equiv-categories TERM($\mathcal{C}_0$) and DIAGR($\mathcal{C}_0$) are equivalent. Intuitively, this means that these equiv-categories have the same equivalence classes of isomorphic objects.

Definition 41 (Equivalence of equiv-categories) Two equiv-categories $\mathcal{C}$ and $\mathcal{C}'$ are equivalent if and only if there exist two equiv-functors $F : \mathcal{C} \to \mathcal{C}'$ and $G : \mathcal{C}' \to \mathcal{C}$ such that $F \circ G \cong Id_{\mathcal{C}'}$ and $G \circ F \cong Id_{\mathcal{C}}$.

Lemma 42 *There exists an equiv-functor $\mathcal{T} : \mathrm{DIAGR}(\mathcal{C}_0) \to \mathrm{TERM}(\mathcal{C}_0)$, which preserves finite equiv-colimits, such that $\mathcal{T} \circ I \cong J$.*

Proof. From theorem 38, the equiv-category $\mathrm{TERM}(\mathcal{C}_0)$ is finitely cocomplete. So the result is immediate from theorem 31. □

Lemma 43 *We have the following natural isomorphisms:*

1. $\mathcal{D} \circ \mathcal{T} \circ I \cong I$;
2. $\mathcal{T} \circ \mathcal{D} \circ J \cong J$.

Proof. This is an immediate consequence of $\mathcal{T} \circ I \cong J$ and $\mathcal{D} \circ J = I$. □

Proposition 44 *There is a natural isomorphism $\mathcal{D} \circ \mathcal{T} \cong Id_{\mathrm{DIAGR}(\mathcal{C}_0)}$.*

Proof. From lemma 43.1, we have $\mathcal{D} \circ \mathcal{T} \circ I \cong I$. Moreover the equiv-functors $\mathcal{D} \circ \mathcal{T}$ and $Id_{\mathrm{DIAGR}(\mathcal{C}_0)}$ preserve finite equiv-colimits. Therefore, from theorem 31, $\mathcal{D} \circ \mathcal{T} \cong Id_{\mathrm{DIAGR}(\mathcal{C}_0)}$. □

Proposition 45 *There is a natural isomorphism $\mathcal{T} \circ \mathcal{D} \cong Id_{\mathrm{TERM}(\mathcal{C}_0)}$.*

Proof. We construct a natural transformation $\Psi : \mathcal{T} \circ \mathcal{D} \dot{\to} Id_{\mathrm{TERM}(\mathcal{C}_0)}$. For every object U of $\mathrm{TERM}(\mathcal{C}_0)$, we define by induction on the structure of U an isomorphism $\Psi_U : \mathcal{T}(\mathcal{D}(U)) \to U$, and, in parallel, we show that Ψ is a natural transformation by induction on the structure of arrows of $\mathrm{TERM}(\mathcal{C}_0)$.

1. $U = J(A)$, where A is an object of $\mathcal{C}_0$. From lemma 43.2, $\mathcal{T} \circ \mathcal{D} \circ J \cong J$. Therefore, there exists an isomorphism $\Psi_U : \mathcal{T}(\mathcal{D}(U)) \to U$.
2. $U = \emptyset$. $\mathcal{T}(\mathcal{D}(\emptyset)) = \mathcal{T}(\bigcirc) \cong \emptyset$ because $\mathcal{T}$ preserves the initial object. Let $\Psi_\emptyset : \mathcal{T}(\mathcal{D}(\emptyset)) \to \emptyset$ be this isomorphism.
3. $U = \mathsf{push}(A, B, C, f, g)$. By induction hypothesis, we have three isomorphisms $\Psi_A : \mathcal{T}(\mathcal{D}(A)) \to A$, $\Psi_B : \mathcal{T}(\mathcal{D}(B)) \to B$ and $\Psi_C : \mathcal{T}(\mathcal{D}(C)) \to C$ such that $f \circ \Psi_A = \Psi_B \circ \mathcal{T}(\mathcal{D}(f))$ and $g \circ \Psi_A = \Psi_C \circ \mathcal{T}(\mathcal{D}(g))$. As $\mathcal{T}$ and $\mathcal{D}$ preserve pushouts, the triple

$$(\mathcal{T}(\mathcal{D}(\mathsf{push}(A,B,C,f,g)),\ \mathcal{T}(\mathcal{D}(\mathsf{k}_1(A,B,C,f,g)),\ \mathcal{T}(\mathcal{D}(\mathsf{k}_2(A,B,C,f,g)))$$

is a pushout of the diagram

$$\mathcal{P}ush\mathcal{D}iagr(\mathcal{T}(\mathcal{D}(A)), \mathcal{T}(\mathcal{D}(B)), \mathcal{T}(\mathcal{D}(C)), \mathcal{T}(\mathcal{D}(f)), \mathcal{T}(\mathcal{D}(g))).$$

Therefore, there is a mediating arrow

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

unique up to equivalence, which is an isomorphism, and such that

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

It remains to show that Ψ is a natural transformation, which is done by induction on the structure of arrows of $\mathrm{TERM}(\mathcal{C}_0)$ (rules 4–10).

□

From propositions 44 and 45, we deduce the following theorem:

Theorem 46 *The equiv-categories* $\mathrm{TERM}(\mathcal{C}_0)$ *and* $\mathrm{DIAGR}(\mathcal{C}_0)$ *are equivalent.*

Intuitively, the equiv-categories $\mathrm{TERM}(\mathcal{C}_0)$ and $\mathrm{DIAGR}(\mathcal{C}_0)$ have the same classes of isomorphic objects. This implies that two objects of $\mathrm{TERM}(\mathcal{C}_0)$ (representing two modular specifications) are isomorphic if and only if their associated diagrams are isomorphic in $\mathrm{DIAGR}(\mathcal{C}_0)$, and that two arrows of $\mathrm{TERM}(\mathcal{C}_0)$ (representing two specification morphisms) are equivalent if and only if their associated diagram morphisms are equivalent in $\mathrm{DIAGR}(\mathcal{C}_0)$.

We have shown that the equiv-categories $\mathrm{TERM}(\mathcal{C}_0)$ and $\mathrm{DIAGR}(\mathcal{C}_0)$ are equivalent. However, we must note that $\mathrm{TERM}(\mathcal{C}_0)$ and $\mathrm{DIAGR}(\mathcal{C}_0)$ *are not isomorphic*, because they have not the same choices of equiv-colimits. For instance, $\mathit{push}(A, B, C, f, g)$ and $\mathit{push}(A, C, B, g, f)$ are two different equiv-pushouts in the equiv-category $\mathrm{TERM}(\mathcal{C}_0)$ which are sent to the same diagram in $\mathrm{DIAGR}(\mathcal{C}_0)$.

Two equiv-categories $\mathcal{C}$ and $\mathcal{C}'$ are equivalent if and only if their corresponding quotient categories $\mathcal{C}/\!\sim$ and $\mathcal{C}'/\!\sim$ are equivalent. Therefore, from theorem 46, we can deduce that the categories $\mathrm{Term}(\mathcal{C}_0)$ and $\mathrm{Diagr}(\mathcal{C}_0)$ are equivalent.

6 Normalization of Diagrams

We have seen that two modular specifications are isomorphic in $\mathrm{TERM}(\mathcal{C}_0)$ if and only if their associated diagrams are isomorphic. However, two isomorphic diagrams need not be *identical.* For example, $\overline{\delta_1}$, $\overline{\delta_2}$, $\overline{\delta_3}$ and $\overline{\delta_4}$ are different diagrams which are isomorphic in $\mathrm{DIAGR}(\mathcal{C}_0)$.

In this section, we propose a normalization of diagrams and show that two diagrams are isomorphic if and only if they have the same normal form. We will suppose that the base category $\mathcal{C}_0$ is finite and cycle free, because we have not solved the problem in the general case.

Definition 47 (Skeletal (equiv-)category) An (equiv-)category $\mathcal{C}$ is skeletal if for all objects A and B of $\mathcal{C}$, $A \cong B \Rightarrow A = B$.

Any category $\mathcal{C}_0$ has an equivalent skeletal category $\mathcal{C}_0'$ which may be constructed by choosing a representative in every equivalence class of isomorphic objects. Given an object A, let $\mathrm{Sk}(A)$ be the chosen object for the class of objects isomorphic to A, and $\phi_A : A \to \mathrm{Sk}(A)$ the corresponding isomorphism. Then, there is a functor $\mathrm{Sk} : \mathcal{C}_0 \to \mathcal{C}_0'$ defined as follows.

$$\begin{array}{cccc} \mathrm{Sk} \;: & \mathcal{C}_0 & \to & \mathcal{C}_0' \\ & A & \mapsto & \mathrm{Sk}(A) \\ & f : A \to B & \mapsto & \phi_B \circ f \circ \phi_A^{-1} \end{array}$$

For any diagram $\overline{\delta} = (\delta^\Phi,\ \delta : \mathcal{P}(\delta^\Phi) \to \mathcal{C}_0)$ over $\mathcal{C}_0$, $\mathrm{Sk} \circ \overline{\delta} = (\delta^\Phi,\ \mathrm{Sk} \circ \delta : \mathcal{P}(\delta^\Phi) \to \mathcal{C}_0')$ is a diagram over $\mathcal{C}_0'$ which is isomorphic to $\overline{\delta}$.

6.1 Hypothesis

For the rest of the paper, we make the following assumptions.

1. The category $\mathcal{C}_0$ is finite i.e. the set of objects and the set of arrows of $\mathcal{C}_0$ is finite.

2. The category $\mathcal{C}_0$ is cycle free, i.e. for every arrow $f : A \to A$ of $\mathcal{C}_0$, $f = id_A$.

3. The category $\mathcal{C}_0$ is skeletal.

The hypothesis that $\mathcal{C}_0$ is skeletal is natural since we need a normal form for every object of $\mathcal{C}_0$ if we want to get a chance to have a normal form for diagrams over $\mathcal{C}_0$.

Lemma 48 *Let A, B be two objects, and $f : A \to B$, $g : B \to A$ be two arrows of $\mathcal{C}_0$. Then, $A = B$ and $f = g = id_A$.*

Proof. Straightforward from hypothesis 2 and 3. □

6.2 Completion of diagrams

We first define *complete* diagrams and show that a diagram may be *completed*, i.e. transformed into an isomorphic complete diagram.

Definition 49 (Complete diagram) A diagram $\overline{\delta}$ is *complete* if

- $\overline{\delta}$ has no edge labeled by an identity arrow: $\forall a : m \to n,\ \delta(a) \neq id_{\delta(m)}$;
- $\overline{\delta}$ contains no couple of edges with same source and target which are labeled by the same arrow: $\forall a_1, a_2 : m \to n \in \text{Edges}(\delta^\Phi),\ \delta(a_1) \neq \delta(a_2)$;
- $\overline{\delta}$ contains all compositions: $\forall a_1 : n_0 \to n_1, a_2 : n_1 \to n_2 \in \text{Edges}(\delta^\Phi)$,
$$\exists a : n_0 \to n_2 \in \text{Edges}(\delta^\Phi)\,;\ \delta(a) = \delta(a_2) \circ \delta(a_1);$$
- $\overline{\delta}$ contains all right factorizations: $\forall a_1 : n_1 \to n_0,\ a_2 : n_2 \to n_0 \in \text{Edges}(\delta^\Phi)$, if $\exists h : \delta(n_1) \to \delta(n_2)$ in $\mathcal{C}_0$ such that $\delta(a_2) \circ h = \delta(a_1)$, then
$$\exists a : n_1 \to n_2 \in \text{Edges}(\delta^\Phi)\,;\ \delta(a) = h.$$

Proposition 50 (Completion) *For every diagram $\overline{\delta}$, there exists a complete diagram*

$$\text{Complete}(\overline{\delta})$$

such that $\overline{\delta} \cong$ Complete$(\overline{\delta})$.

Proof. From $\overline{\delta}$, we construct a new diagram by iterating the following transformations.

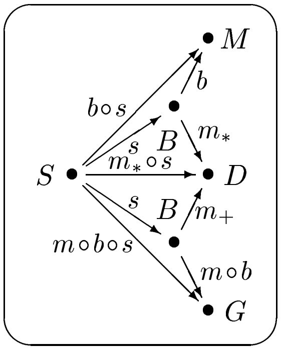
1. For every $a : m \to n \in \text{Edges}(\delta^\Phi)$ such that $\delta(m) = \delta(n)$ and $\delta(a) = id_{\delta(m)}$, we remove the edge a of $\overline{\delta}$. If $m \neq n$, then we also merge m and n.
2. For all $a_1, a_2 : m \to n \in \text{Edges}(\delta^\Phi)$ such that $\delta(a_1) = \delta(a_2)$, we remove the edge a_2.
3. For all $\forall a_1 : n_0 \to n_1, a_2 : n_1 \to n_2 \in \text{Edges}(\delta^\Phi)$, if there is no edge $a : n_0 \to n_2 \in \text{Edges}(\delta^\Phi)$ labeled by $\delta(a_2) \circ \delta(a_1)$, we add such an edge in $\overline{\delta}$.
4. $\forall a_1 : n_1 \to n_0, a_2 : n_2 \to n_0 \in \text{Edges}(\delta^\Phi)$, if there exists $h : \delta(a_1) \to \delta(a_2)$ in $\mathcal{C}_0$ and if there is no edge $a : n_1 \to n_2 \in \text{Edges}(\delta^\Phi)$ labeled by h, we add such an edge in $\overline{\delta}$.

We can check that

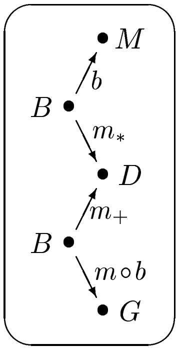
- this procedure stops, because the category $\mathcal{C}_0$ is finite;
- every transformation yields an isomorphic diagram.

Therefore, we end up with a complete diagram Complete$(\overline{\delta})$ which is isomorphic to $\overline{\delta}$. □

For example, $\text{Complete}(\overline{\delta_1}) = \text{Complete}(\overline{\delta_3}) = \text{Complete}(\overline{\delta_4}) = \overline{\delta'}$. Therefore, $\overline{\delta_1} \cong \overline{\delta_3} \cong \overline{\delta_4}$. The diagram $\overline{\delta_2}$ is complete: $\text{Complete}(\overline{\delta_2}) = \overline{\delta_2}$.



$\overline{\delta'}$



$\overline{\delta_2}$

6.3 Normalization

Completing a diagram allows us to get a more "canonical" form for diagrams. However, two complete diagrams may be isomorphic without being identical e.g. $\overline{\delta_2}$ and $\overline{\delta'}$.

Definition 51 (Elementary zigzag) An *elementary zigzag* of a graph δ^Φ is a zigzag

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

of δ^Φ such that a_0 and a_1 are distinct edges.

Intuitively, a zigzag Z is included in a zigzag Z' if Z is a "sub-zigzag" of Z'.

Definition 52 (Inclusion of zigzag)
Let $Z = (k, Z_V, Z_E)$ and $Z' = (k', Z'_V, Z'_E)$ be two zigzags of a graph δ^Φ, with

$$\begin{array}{l} Z_V = (n_0, n_1, \ldots, n_k) \ \text{ and } \ Z_E = (a_0, a_1, \ldots, a_{k-1})\,; \\ Z'_V = (n'_0, n'_1, \ldots, n'_{k'}) \ \text{ and } \ Z'_E = (a'_0, a'_1, \ldots, a'_{k'-1}). \end{array}$$

$Z \subseteq Z'$ if there exists an integer j, $0 \le j \le k' - k$, such that

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

Note that $Z \subseteq Z'$ implies $k \le k'$.

Let $\overline{\delta} = (\delta^\Phi,\ \delta : \mathcal{P}(\delta^\Phi) \to \mathcal{C}_0)$ be a diagram. Elementary zigzags of $\overline{\delta}$ are elementary zigzags of δ^Φ. We now define an ordering on elementary zigzags of a *complete diagram.*

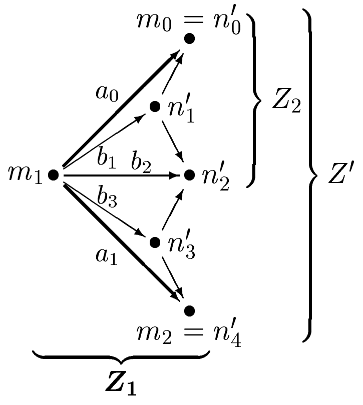
Definition 53 Let $\overline{\delta}$ be a complete diagram over $\mathcal{C}_0$. We define an ordering $\le$ on elementary zigzags of $\overline{\delta}$ as follows.

Let Z_1 and Z_2 be two elementary zigzags. Let $Z_1 = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} m_2$.

- We have $Z_1 < Z_2$ if there exists a zigzag $Z' = (k', Z'_V, Z'_E)$, with $Z'_V = (n'_0, n'_1, \ldots, n'_{k'})$ such that

- $m_0 = n'_0$ and $m_2 = n'_{k'}$;
- $Z_2 \subseteq Z'$;
- $\delta(a_0) \sim_{\overline{\delta}} \delta(a_1)\ [Z']$;
- $\forall i,\ 0 \leq i \leq k',\ m_1 \neq n'_i$.

– We have $Z_1 \leq Z_2$ if $Z_1 = Z_2$ or $Z_1 < Z_2$.



Proposition 54 *The relation $\leq$ is an ordering.*

For example, in the diagram $\overline{\delta'}$, we have

$$\begin{array}{lcl}
M \xleftarrow{b\circ s} S \xrightarrow{m\circ b\circ s} G & \leq & M \xleftarrow{b} B \xrightarrow{m_*} D, \\
M \xleftarrow{b\circ s} S \xrightarrow{m\circ b\circ s} G & \leq & D \xleftarrow{m_+} B \xrightarrow{m\circ b} G, \\
M \xleftarrow{b\circ s} S \xrightarrow{m_*\circ s} D & \leq & M \xleftarrow{b} B \xrightarrow{m_*} D, \\
D \xleftarrow{m_*\circ s} S \xrightarrow{m\circ b\circ s} G & \leq & D \xleftarrow{m_+} B \xrightarrow{m\circ b} G.
\end{array}$$

The set of elementary zigzags of a diagram $\overline{\delta}$ is finite, therefore it contains *maximal elements*. For example, the elementary zigzags

$$M \xleftarrow{b} B \xrightarrow{m_*} D, \quad D \xleftarrow{m_+} B \xrightarrow{m\circ b} G$$

are maximal in $\overline{\delta'}$. But

$$M \xleftarrow{b\circ s} S \xrightarrow{m\circ b\circ s} G, \quad M \xleftarrow{b\circ s} S \xrightarrow{m_*\circ s} D, \quad D \xleftarrow{m_*\circ s} S \xrightarrow{m\circ b\circ s} G$$

are not.

Definition 55 (Terminating vertex) A *terminating vertex* of a graph δ^Φ is a vertex n such that there is no edge with source n in δ^Φ.

Definition 56 (Link) A *link* of a complete diagram $\overline{\delta}$ is a maximal elementary zigzag of $\overline{\delta}$

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

such that m_0 and m_2 are terminating vertices of δ^Φ.

The set of links of a complete diagram is computable, because the set of elementary zigzags is finite, and the ordering is decidable.

Lemma 57 *Let $\overline{\delta}$ be a complete diagram, n, n' be two vertices of δ^Φ, and $u : A \to \delta(n)$, $v : A \to \delta(n')$ be two arrows of $\mathcal{C}_0$. Then,*

$$u \sim_{\overline{\delta}} v\ [Z] \ \Rightarrow\ u \sim_{\overline{\delta}} v\ [Z']$$

where $Z' = m_0 \xleftarrow{a_0} m_1 \xrightarrow{a_1} \dots \xleftarrow{a_{k-2}} m_{k-1} \xrightarrow{a_{k-1}} m_k$ is a zigzag such that for each i verifying $0 \leq i \leq (k-2)/2$, we have $a_{2i} : m_{2i+1} \to m_{2i}$, $a_{2i+1} : m_{2i+1} \to m_{2i+2}$ and $m_{2i+1} \xleftarrow{a_{2i}} m_{2i+1} \xrightarrow{a_{2i+1}} m_{2i+2}$ is a link of $\overline{\delta}$.

Definition 58 (Normal form) A diagram $\overline{\delta}$ is in *normal form* if

- every vertex of δ^Φ is a terminating vertex, or belongs to a link of $\overline{\delta}$;
- every edge of δ^Φ belongs to a link of $\overline{\delta}$.

Proposition 59 (Normalization) *For every diagram $\overline{\delta}$, there exists a diagram $\mathcal{N}(\overline{\delta})$ in normal form such that $\overline{\delta} \cong \mathcal{N}(\overline{\delta})$.*

Proof. We consider the set

$$E := \{a \in \text{Edges}(\overline{\delta}),\ a \text{ does not belong to a link of } \overline{\delta}\}.$$

$\mathcal{N}(\overline{\delta})$ is the diagram which is obtained by removing all edges of E from $\overline{\delta}$. We prove that we indeed get a diagram which is isomorphic to $\overline{\delta}$ by induction on $\text{Card}(E)$. The important point is to "suppress edges in $\overline{\delta}$ in the right order".

Base step. If $E = \emptyset$, then $\overline{\delta}$ is in normal form.

Inductive step. We construct a diagram $\overline{\delta'}$ by removing an edge a of E from $\overline{\delta}$. Let m be a vertex of δ^Φ such that there exists an edge $a_0 : m \to n_0 \in E$. Such a vertex exists because δ^Φ has no cycle. Then, let $F = \{a_i : m \to n_i,\ a_i \in E\} \subseteq E$. We have of course $F \neq \emptyset$.

1. If $\text{Card}(F) = 1$, we have a unique edge $a : m \to n$ in F. We construct a diagram $\overline{\delta'}$ by removing the vertex m and the edge a from $\overline{\delta}$.

   We have an isomorphism $\overline{\sigma} : \overline{\delta} \to \overline{\delta'}$ such that $\sigma^\Phi(m) = n$, $\sigma^\Phi(a) = 0_n$ and $\sigma_m = \delta(a)$.

2. If $\text{Card}(F) \leq 2$ and if there exists in F an edge $a : m \to n$ such that n *is not a terminating vertex*, then we construct a diagram $\overline{\delta'}$ from $\overline{\delta}$ by removing the edge a. There exist two edges

   $$a_1 : n \to n_1,\ a_2 : m \to n_1 \in \text{Edges}(\delta^\Phi)$$

   such that $\delta(a_1) \circ \delta(a) = \delta(a_2)$
   We have an isomorphism $\overline{\sigma} : \overline{\delta} \to \overline{\delta'}$ such that $\sigma^\Phi(a) = m \xrightarrow{a_2} n_1 \xleftarrow{a_1} n$.

3. If $\text{Card}(F) \leq 2$ and if every edge $a_i : m \to n_i$ is such that n is a terminating vertex, let $a : m \to n, a' : m \to n' \in F$. As the elementary zigzag

   $$n \xleftarrow{a} m \xrightarrow{a'} n'$$

   is not maximal, there exists a zigzag $Z : n \rightsquigarrow n'$ in $\overline{\delta}$, which only contains links (lemma 57), and such that $\delta(a) \sim_{\overline{\delta}} \delta(a')\ [Z]$. Then, we construct a diagram $\overline{\delta'}$ from $\overline{\delta}$ by removing the edge a.

   We have an isomorphism $\overline{\sigma} : \overline{\delta} \to \overline{\delta'}$ such that $\sigma^\Phi(a) = m \xrightarrow{a'} n' \overset{Z}{\leftsquigarrow} n$.

□

Corollary 60 *Any two diagrams having the same normal form are isomorphic.*

Back to the example, we have $\mathcal{N}(\overline{\delta'}) = \overline{\delta_2}$ and $\mathcal{N}(\overline{\delta_2}) = \overline{\delta_2}$. Therefore, $\overline{\delta'} \cong \overline{\delta_2}$, hence $\overline{\delta_1} \cong \overline{\delta_2} \cong \overline{\delta_3} \cong \overline{\delta_4}$.

Theorem 61 *Let $\overline{\alpha}$ and $\overline{\beta}$ be two diagrams. Then, $\overline{\alpha} \cong \overline{\beta} \iff \mathcal{N}(\overline{\alpha}) = \mathcal{N}(\overline{\beta})$.*

Proof. The $\Leftarrow$ part comes from corollary 60. It remains to show the $\Rightarrow$ part. We can suppose w.l.o.g. that $\overline{\alpha}$ and $\overline{\beta}$ are complete. Let $\overline{\sigma} : \overline{\alpha} \to \overline{\beta}$ be an isomorphism in DIAGR($\mathcal{C}_0$).

1. We show that for any terminating vertex n of α^Φ, $\sigma^\Phi(n)$ is a terminating vertex of β^Φ, $\alpha(n) = \beta(\sigma^\Phi(n))$ and $\sigma_n = id_{\alpha(n)}$.

2. Let

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

be a link of $\overline{\alpha}$. Let $n_0 = \sigma^\Phi(m_0)$ and $n_2 = \sigma^\Phi(m_2)$. Thus, the vertices n_0 and n_2 are terminating vertices of β^Φ and

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

Then, we prove that there exists a vertex n_1 in β^Φ such that $\alpha(m_1) = \beta(n_1)$. At last, we show that there exist two edges $b_0 : n_1 \to n_0$ and $b_1 : n_1 \to n_2$ in β^Φ such that $\alpha(a_0) = \beta(b_0)$ and $\alpha(a_1) = \beta(b_1)$.

Eventually, the diagrams $\overline{\alpha}$ and $\overline{\beta}$ have the same terminating vertices and the same links. Therefore, $\mathcal{N}(\overline{\alpha}) = \mathcal{N}(\overline{\beta})$. □

7 Conclusion

In this paper, we proposed a theoretical framework to study modular specifications. We revisited a classic idea in algebraic specification, which consists in modeling the composition of modular specifications by means of colimits of diagrams.

We proposed a term language to represent specifications built from a category of base specifications $\mathcal{C}_0$ with pushout constructions. This language is formally characterized by a finitely cocomplete equiv-category TERM($\mathcal{C}_0$), which is freely generated over $\mathcal{C}_0$ by a chosen equiv-initial object and chosen equiv-pushouts.

We proposed to represent terms denoting modular specifications as diagrams and thus defined an equiv-category DIAGR($\mathcal{C}_0$) of diagrams. This equiv-category is a completion of $\mathcal{C}_0$ by finite equiv-colimits. The association of terms with diagrams is described by an equiv-functor $\mathcal{D}$: TERM($\mathcal{C}_0$) $\to$ DIAGR($\mathcal{C}_0$) which defines an equivalence between the equiv-categories TERM($\mathcal{C}_0$) and DIAGR($\mathcal{C}_0$).

We made a careful distinction between syntactic entities, i.e. objects and arrows in an *equiv-category*, which may be handled effectively, and their meaning in the corresponding *category*. This effective representation of terms in TERM($\mathcal{C}_0$), and of diagrams in DIAGR($\mathcal{C}_0$), allowed us to define an equivalence between both equiv-categories as a computable equiv-functor $\mathcal{D}$.

At last, we gave a procedure to compute the normal form of a diagram, in the case when the base category is skeletal, finite and cycle free. In this case, we can thus decide whether two specifications are related by a construction isomorphism by comparing the normal form of their associated diagrams.

Future Work

We have proposed a normalization of diagrams when the base category is finite and cycle free. We do not know whether the problem of diagram isomorphism is, in the general case, decidable.

At the present time, it is not possible to specify distinguished cones in the base category, as in sketch theory [14, 11, 12]. It would be interesting to extend our work in order to take into account some pushouts in the base category. This extension would provide us with a finer structure of the library of base specifications.

From a practical point of view, our work could be applied to design or enhance specification and programming languages. Applying the representation of module composition with diagrams, we hope to get a more general concept of modularity, which supports renaming of types and functions as well as consistent sharing of modules.

Moreover, one interest of modularity in programming is to be able to reuse modules in different contexts. The idea here is to have a library of base specifications together with various modules implementing them. Then, the diagram associated with a modular specification describes all the sharing constraints between the imported modules, and could be used to check the compatibility between these modules.

Acknowledgments

I wish to thank Didier Bert for supervising this research, and Jean-Claude Reynaud, whom I borrowed the initial ideas of this work, for numerous helpful discussions. I am also grateful to Christian Lair for pointing out the importance of equiv-categories, and for explaining to me that the category Term($\mathcal{C}_0$) was *not* freely generated over $\mathcal{C}_0$ by a chosen initial object and chosen pushouts.

References

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

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

[3] D. Bert, R. Echahed, P. Jacquet, M.-L. Potet, and J.-C. Reynaud. Spécification, généricité, prototypage : aspects du langage LPG. *Technique et science informatique*, 14(9):1097–1129, 1995.

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

[5] D. Bert and C. Oriat. A model inference system for generic specification with application to code sharing. In *Proceedings of TAPSOFT'95*, number 915 in LNCS, pages 741–755. Springer-Verlag, 1995.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

[14] C. Ehresmann. Esquisses et types de structures algébriques. *Bulletin de l'Institut Polytechnique de Iaşi*, 14(1), 1968.

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

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

[26] 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.

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

[28] C. Oriat. Étude des spécifications modulaires : constructions de colimites finies, diagrammes, isomorphismes. Thèse de doctorat, INPG, Grenoble, Janvier 1996. Anonymous ftp at `ftp.imag.fr`, in `pub/Mediatheque.IMAG/theses/96-Oriat.Catherine/`.

[29] J.-C. Reynaud. Sémantique de LPG. Research Report 651 I IMAG, LIFIA, Mars 1987.

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

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

[32] J.-C. Reynaud. Isomorphism of typed algebraic specifications. Research Report, LGI-IMAG, April 1993.

[33] 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.

[34] 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.

[35] W. Tholen and A. Tozzi. Completions of categories and initial completions. *Cahiers de Topologie et Géométrie Différentielle Catégorique*, 30:127–156, 1989.

[36] 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.

[37] 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.