Proceedings of AMAST'95, Fourth International Conference on Algebraic Methodology and Software Technology, Montréal, July 1995. LNCS 936, pages 184–198, 1995.

Detecting Isomorphisms of Modular Specifications with Diagrams

Catherine Oriat

LGI – IMAG, BP 53, 38041 Grenoble Cedex 9, France
e-mail: Catherine.Oriat@imag.fr

Abstract. We propose to detect isomorphisms of algebraic modular specifications, by representing specifications as diagrams over a category $\mathcal{C}_0$ of base specifications and specification morphisms. We start with a formulation of modular specifications as terms, which are interpreted as diagrams. This representation has the advantage of being more abstract, i.e. less dependent of one specific construction than terms. For that, we define a category $\mathrm{diagr}\,(\mathcal{C}_0)$ of diagrams, which is a completion of $\mathcal{C}_0$ with finite colimits. The category $\mathrm{diagr}\,(\mathcal{C}_0)$ is *finitely cocomplete*, even if $\mathcal{C}_0$ is *not* finitely cocomplete. We define a functor $\mathcal{D}[\![]\!] : \mathrm{Term}\,(\mathcal{C}_0) \to \mathrm{diagr}\,(\mathcal{C}_0)$ which maps specifications to diagrams, and specification morphisms to diagram morphisms. This interpretation is sound in that the colimit of a diagram representing a specification is isomorphic to this specification. The problem of isomorphisms of modular specifications is solved by detecting isomorphisms of diagrams.

1 Introduction

The specification of large systems requires the use of modular specifications. Small specifications are combined in a structured way to construct larger specifications. In this paper, we only consider specifications which are built with colimits over a category $\mathcal{C}_0$ of fixed base specifications. Formally, this means that we work in the category freely generated by finite colimits over $\mathcal{C}_0$. If we wish to add a new specification a posteriori in the category (for instance with an enrichment or hiding), we need to work in a new category which contains the already constructed specifications as well as the new defined specification.

The aim of this paper is to solve the problem of isomorphism of modular specifications in this setting of constructions with finite colimits. There are several kinds of isomorphisms we could be interested in:

- "Isomorphism of name". Two specifications are isomorphic if they have the same name. This simple definition provides a very weak form of isomorphism.
- "Isomorphism of structure". 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.

- "Semantic isomorphism". Two specifications are isomorphic if their associated classes of models (defined by the semantics) are the same. This isomorphism is very hard to treat, mainly because classes of algebras cannot be manipulated easily.
- "Isomorphism in SPEC". Two specifications are isomorphic if they are related by a bijective specification morphism. The difficulty here is first to exhibit this morphism, and secondly to check that it is indeed a *specification* morphism, i.e. that the equations of each specification hold in the other one.
- The isomorphisms we consider here are "construction isomorphisms". Two specifications are isomorphic if we can prove it using the general properties of colimits. We think this kind of isomorphism is interesting because it not too general in that it reflects the constructions of the modular specification. But it is more general than the isomorphism of structure because the specific steps chosen for the construction are abstracted. These isomorphisms do not depend on the actual definition of the base specifications, they only depend on how base specifications, related by base specification morphisms, are combined. Of course if two base specifications are isomorphic, and if this isomorphism is not part of $\mathcal{C}_0$, then we will fail to find it.

Most existing specification languages give more importance to the construction of a modular specification than to the result of the construction; for example CLEAR [3, 4], ACTONE [6], ASL [14], OBJ2 [7], PLUSS [8, 2], LPG [1]. This implies that the only tractable isomorphisms are isomorphisms of structure. We propose to adopt a less syntactic view of modular specification, by representing them as diagrams over the category of base specifications $\mathcal{C}_0$. This representation is more abstract than terms because irrelevant steps of the construction disappear. We need of course to work in a *category* of diagrams, and so we associate specification morphisms between modular specifications to diagram morphisms. This approach is similar to that adopted to describe the semantics of CLEAR [4]. Our diagrams correspond to based theories. We need a more general definition for arrows than that of based morphisms, because based morphisms only correspond to inclusions of modular specifications, whereas we want a diagram morphism to correspond to any specification morphism. So we define a category of diagrams $\mathrm{diagr}\,(\mathcal{C}_0)$, and "construction isomorphisms" of modular specifications then correspond to isomorphisms in the category $\mathrm{diagr}\,(\mathcal{C}_0)$.

This paper is organized as follows. In section 2, we present the category $\mathrm{Term}\,(\mathcal{C}_0)$, which provides a syntax for modular specifications and specification morphisms. In section 3, we present examples of modular specifications to illustrate the syntax and motivate the definition of a category $\mathrm{DIAGR}\,(\mathcal{C}_0)$. In section 4, we define the categories $\mathrm{DIAGR}\,(\mathcal{C}_0)$ and $\mathrm{diagr}\,(\mathcal{C}_0)$ and present some theoretical results. In section 5, we explain how terms denoting specifications or specification morphisms can be associated to diagrams and diagram morphisms. This mapping is described by a functor $\mathcal{D}[\![]\!] : \mathrm{Term}\,(\mathcal{C}_0) \to \mathrm{diagr}\,(\mathcal{C}_0)$. In section 6, we present an algorithm to detect when two diagrams are isomorphic in the category $\mathrm{diagr}\,(\mathcal{C}_0)$, when the base category $\mathcal{C}_0$ is finite and has no cycle.

2 Syntax for Modular Specifications

In this section, we present a syntax for modular algebraic specifications constructed with colimits. This syntax is formulated with the concept of *dependent types* as suggested by Cartmell [5]. Cartmell's generalized algebraic theories are a generalization of many-sorted algebras, which allow to define dependent types, i.e. types parameterized by terms. This approach has already been presented in [10, 11, 12]. We suppose we have a category $\mathcal{C}_0$ of base specifications and specification morphisms.

We have two types: the type of specifications Spec, and the type of specification morphisms Hom, which depends on two specifications. If A and B are specifications, $\mathsf{Hom}\,(A, B)$ is the type of specification morphisms from A to B.

$$\frac{}{\mathsf{Spec}\text{ is a type}} \qquad \frac{A, B : \mathsf{Spec}}{\mathsf{Hom}\,(A, B)\text{ is a type}} \tag{1, 2}$$

We now define *terms* of both types Spec and Hom, as well as axioms satisfied by these terms.

$$\frac{Sp \text{ specification of } \mathcal{C}_0}{Sp : \mathsf{Spec}} \qquad \frac{p : Sp_1 \to Sp_2 \text{ specification morphism of } \mathcal{C}_0}{p : \mathsf{Hom}\,(Sp_1, Sp_2)} \tag{3, 4}$$

$$\frac{A, B, C : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(B, C)}{g \circ f : \mathsf{Hom}\,(A, C)} \tag{5}$$

$$\frac{A, B, C, D : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(B, C) \;\;;\;\; h : \mathsf{Hom}\,(C, D)}{(h \circ g) \circ f = h \circ (g \circ f) : \mathsf{Hom}\,(A, D)} \tag{6}$$

$$\frac{A : \mathsf{Spec}}{\mathsf{id}_A : \mathsf{Hom}\,(A, A)} \tag{7}$$

$$\frac{A, B : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B)}{f \circ \mathsf{id}_A = f : \mathsf{Hom}\,(A, B)} \qquad \frac{A, B : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B)}{\mathsf{id}_B \circ f = f : \mathsf{Hom}\,(A, B)} \tag{8, 9}$$

An algebra which satisfies the generalized algebraic theory specified by rules (1)–(2) and (5)–(9) is a category. We now give a syntax for colimit constructions. Here, for instance we give a syntax for an initial object and pushouts.

$$\frac{}{\emptyset : \mathsf{Spec}} \qquad \frac{A : \mathsf{Spec}}{\mathsf{j}_A : \mathsf{Hom}\,(\emptyset, A)} \qquad \frac{A : \mathsf{Spec} \;\;;\;\; f, g : \mathsf{Hom}\,(\emptyset, A)}{f = g : \mathsf{Hom}\,(\emptyset, A)} \tag{10, 11, 12}$$

$$\frac{A, B, C : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C)}{\mathsf{push}\,(A, B, C, f, g) : \mathsf{Spec}} \tag{13}$$

$$\frac{A, B, C : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C)}{\&_1(A, B, C, f, g) : \mathsf{Hom}\,(B, \mathsf{push}\,(A, B, C, f, g))} \tag{14}$$

$$\frac{A, B, C : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C)}{\&_2(A, B, C, f, g) : \mathsf{Hom}\,(C, \mathsf{push}\,(A, B, C, f, g))} \tag{15}$$

$$\frac{A, B, C : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C)}{\&_1(A, B, C, f, g) \circ f = \&_2(A, B, C, f, g) \circ g : \mathsf{Hom}\,(A, \mathsf{push}\,(A, B, C, f, g))} \tag{16}$$

$$\frac{\begin{array}{c} A, B, C, D : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C) \\ f' : \mathsf{Hom}\,(B, D) \;\;;\;\; g' : \mathsf{Hom}\,(C, D) \;\;;\;\; f' \circ f = g' \circ g : \mathsf{Hom}\,(A, D) \end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') : \mathsf{Hom}\,(\mathsf{push}\,(A, B, C, f, g), D)} \tag{17}$$

$$\frac{\begin{array}{c} A, B, C, D : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C) \\ f' : \mathsf{Hom}\,(B, D) \;\;;\;\; g' : \mathsf{Hom}\,(C, D) \;\;;\;\; f' \circ f = g' \circ g : \mathsf{Hom}\,(A, D) \end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_1(A, B, C, f, g) = f' : \mathsf{Hom}\,(B, D)} \tag{18}$$

$$\frac{\begin{array}{c} A, B, C, D : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C) \\ f' : \mathsf{Hom}\,(B, D) \;\;;\;\; g' : \mathsf{Hom}\,(C, D) \;\;;\;\; f' \circ f = g' \circ g : \mathsf{Hom}\,(A, D) \end{array}}{\mathsf{up}\,(A, B, C, D, f, g, f', g') \circ \&_2(A, B, C, f, g) = g' : \mathsf{Hom}\,(C, D)} \tag{19}$$

$$\frac{\begin{array}{c} A, B, C, D : \mathsf{Spec} \;\;;\;\; f : \mathsf{Hom}\,(A, B) \;\;;\;\; g : \mathsf{Hom}\,(A, C) \\ u, v : \mathsf{Hom}\,(\mathsf{push}\,(A, B, C, f, g), D) \\ u \circ \&_k(A, B, C, f, g) = v \circ \&_k(A, B, C, f, g) \;\;;\;\; (k = 1, 2) \end{array}}{u = v : \mathsf{Hom}\,(\mathsf{push}\,(A, B, C, f, g), D)} \tag{20}$$

An algebra which satisfies this specification is a finitely cocomplete category.

We must note that this specification is actually not a generalized algebraic theory, because the rules (17)–(20) contain equalities in their premises. To write a proper generalized algebraic theory, one has to axiomatize equality in the type system with a predicate eq [5]. These equalities raise another problem: it may not be decidable whether or not a term is well-formed. A rigorous construction of the category freely generated by a chosen initial object and chosen pushouts as a category of *terms* is under development.

Let $\mathrm{Term}\,(\mathcal{C}_0)$ be the algebra freely generated on the specified colimit constructions. Let SPEC be the category of all specifications, with a *chosen* initial object, and *chosen* pushouts. SPEC is therefore an algebra which satisfies the equations. $\mathcal{C}_0$ is a subcategory of SPEC, and of $\mathrm{Term}\,(\mathcal{C}_0)$. We note these inclusions

$$i : \mathcal{C}_0 \to \mathrm{SPEC}, \quad e : \mathcal{C}_0 \to \mathrm{Term}\,(\mathcal{C}_0)$$

As $\mathrm{Term}\,(\mathcal{C}_0)$ is a free algebra, there exists a unique homomorphism (which is also a functor)

$$\mathcal{S}[\![]\!] : \mathrm{Term}\,(\mathcal{C}_0) \to \mathrm{SPEC}$$

such that $\mathcal{S}[\![]\!] \circ e = i$. This functor associates to each term of type Spec the specification that it represents, and to each term of type Hom the specification morphism it represents. $\mathcal{S}[\![]\!]$ is a "standard semantics" for terms.

3 Example

The aim of this section is first to give some examples of modular specifications written with the syntax presented in the previous section, and secondly to motivate the definition of the category of diagrams presented in the next section.

We present different ways of specifying the theory of rings in the specification language LPG.

$S =$
```
property A-SORT
sorts    s
```

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

S specifies a single sort, B specifies a binary operator and a specification morphism $s : S \to B$, defined by the statement `satisfies A-SORT[s]` in 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]
```

M specifies a monoid, with a specification morphism $b : B \to M$. 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 $m : M \to G$.

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

D specifies two operators related by the distributive law, and two specification morphisms $m_+, m_* : B \to D$. m_+ maps `op` to `+` and m_* maps `op` to `*`.

To summarize what we have defined so far, we work in the category $\mathcal{C}_0$ (Fig. 1).

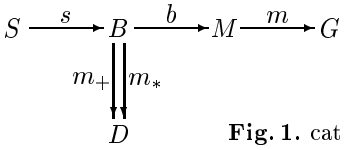


Fig. 1. category $\mathcal{C}_0$

We can now define several modular specifications of rings with pushouts. In LPG, such specifications can be defined with the `combines` construction.

$R_1 = \mathsf{push}\,(B, M, \mathsf{push}\,(B, D, G, m_+, m \circ b), b, \&_1(B, D, G, m_+, m \circ b) \circ m_*)$
$R'_1 = \mathsf{push}\,(B, \mathsf{push}\,(B, M, D, b, m_*), G, \&_2(B, M, D, b, m_*) \circ m_+, m \circ b)$

R_1 and R'_1 are two specifications of rings. Here, the difference is somehow artificially introduced by the syntactic construction push . Indeed both constructions are a coding with pushouts of the colimit of the diagram $\boldsymbol{\delta}_1$ (Fig. 2).

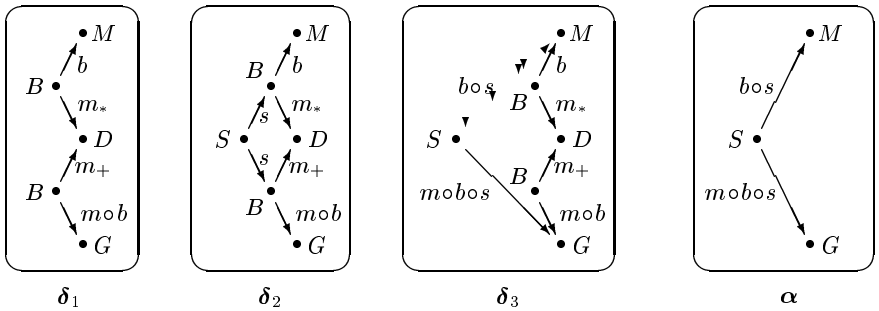


Fig. 2. diagrams $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$, $\boldsymbol{\delta}_3$ and $\boldsymbol{\alpha}$

In the following, if $p = \mathsf{push}\,(x, y, z, u, v)$, we will write $\&_i(p)$ for $\&_i(x, y, z, u, v)$. More complicated cases may arise. We can for instance define a "pseudo ring" i.e. a ring without distributivity either with the term P_3, or P'_3 as follows.

$P_1 = \mathsf{push}\,(S, B, B, s, s)$
$P_2 = \mathsf{push}\,(B, M, P_1, b, \&_1(P_1))$
$P_3 = \mathsf{push}\,(B, P_2, G, \&_2(P_2) \circ \&_2(P_1), m \circ b)$
$P'_3 = \mathsf{push}\,(S, M, G, b \circ s, m \circ b \circ s)$

(P'_3 corresponds to the colimit of the diagram $\boldsymbol{\alpha}$.)

Now we can "add the distributivity" on two different ways and get two new specifications of rings R_2 and R_3.

$$\begin{aligned} R_2 = \mathsf{push}\,(&\mathsf{push}\,(\emptyset, B, B, \mathsf{j}_B, \mathsf{j}_B), D, P_3, \mathsf{up}\,(\emptyset, B, B, D, \mathsf{j}_B, \mathsf{j}_B, m_*, m_+),\\ &\mathsf{up}\,(\emptyset, B, B, P_3, \mathsf{j}_B, \mathsf{j}_B, \&_1(P_3) \circ \&_2(P_2) \circ \&_1(P_1),\\ &\quad \&_1(P_3) \circ \&_2(P_2) \circ \&_2(P_1)))\\ R_3 = \mathsf{push}\,(&\mathsf{push}\,(\emptyset, B, B, \mathsf{j}_B, \mathsf{j}_B), D, P'_3, \mathsf{up}\,(\emptyset, B, B, D, \mathsf{j}_B, \mathsf{j}_B, m_*, m_+),\\ &\mathsf{up}\,(\emptyset, B, B, P'_3, \mathsf{j}_B, \mathsf{j}_B, \&_1(P'_3) \circ b, \&_2(P'_3) \circ m \circ b)) \end{aligned}$$

We will see in section 5 that the specifications R_2 and R_3 correspond to the diagrams $\boldsymbol{\delta}_2$ and $\boldsymbol{\delta}_3$. It is possible to check that the colimits of $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ and $\boldsymbol{\delta}_3$ are all isomorphic. This comes from the equality $m_+ \circ s = m_* \circ s$ in $\mathcal{C}_0$. In other words, the fact that both binary operations are defined on the same set is contained in the distributivity property D.

4 Categories of Diagrams

In the following, we assume the reader is familiar with basic notions of category theory. Vertices(α^Φ) and Edges(α^Φ) respectively denote the set of vertices and the set of edges of a graph α^Φ.

4.1 The Category DIAGR ($\mathcal{C}_0$)

Definition 1 (Diagram). A diagram over a category $\mathcal{C}_0$ is a couple

$$\boldsymbol{\alpha} = (\alpha^\Phi,\ \alpha : \alpha^\Phi \to \mathcal{C}_0),$$

where α^Φ is a graph and $\alpha : \alpha^\Phi \to \mathcal{C}_0$ is a graph morphism. A diagram is *finite* when its underlying graph is finite.

To get a *category* of diagrams, we need to define diagram morphisms. We could consider couples

$$\boldsymbol{\sigma : \alpha \to \beta} = (\sigma^\Phi : \alpha^\Phi \to \beta^\Phi\ ;\ \sigma : \alpha \xrightarrow{\bullet} \beta \circ \sigma^\Phi)$$

where σ^Φ is a graph morphism, and σ a natural transformation. This definition appears in [13] (it is the "flatten" category **Funct** $(\mathcal{C}_0)$, page 244, example 4), and in a dual form in [9] (it is the "super-comma category", page 111, exercise 5.b.) This definition is not general enough, because some specification morphisms have no corresponding diagram morphisms. For instance, there is a morphism

$$\mathsf{up}\,(S, M, G, R_1, b \circ s, m \circ b \circ s, \&_1(R_1), \&_2(R_1) \circ \&_2(B, D, G, m_+, m \circ b))$$

from P_3' to R_1, which corresponds to an arrow from Colim $\boldsymbol{\alpha}$ to Colim $\boldsymbol{\delta}_1$, because $m_+ \circ s = m_* \circ s$. But there is no diagram morphism from $\boldsymbol{\alpha}$ to $\boldsymbol{\delta}_1$ with the definition above (Fig. 3). We need a more general definition of arrows, and thus must consider *generalized graph morphisms*, which associate a *zigzag* to each edge of a graph, and *generalized natural transformations*. With this definition, we can define an arrow $\boldsymbol{\sigma} : \boldsymbol{\alpha} \to \boldsymbol{\delta}_1$, which consists of a generalized graph morphism σ^Φ and a generalized natural transformation σ, defined for instance as follows:
$\sigma^\Phi(1) = 1'$, $\sigma^\Phi(2) = 2'$, $\sigma^\Phi(3) = 3'$; $\sigma^\Phi(a_0) =$ zigzag from $1'$ to $2'$, $\sigma^\Phi(a_1) =$ zigzag from $1'$ to $3'$; $\sigma_1 = m_* \circ s$, $\sigma_2 = id_M$, $\sigma_3 = id_G$.

Definition 2 (Zigzag on a graph). Let α^Φ be a graph. A *zigzag* on α^Φ is a finite linear sequence of edges of α^Φ:

$$Z = n_0 \xrightarrow{a_0} n_1 \xleftarrow{a_1} n_2 \cdots \xrightarrow{a_{k-1}} n_k, \quad \text{noted } Z : n_0 \rightsquigarrow n_k,$$

each edge is oriented either from left to right or from right to left.

We get a graph Zigzag(α^Φ), with the same vertices as α^Φ, and with edges the zigzags of α^Φ.

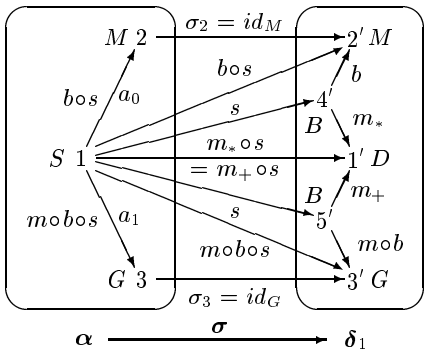


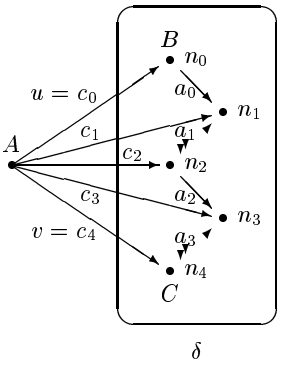
Fig. 3. diagram morphism $\boldsymbol{\sigma : \alpha \to \delta_1}$

Definition 3 (Generalized graph morphism). Let us consider two graphs α^Φ and β^Φ. A *generalized graph morphism* $\sigma^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$ from α^Φ to β^Φ is a graph morphism from α^Φ to Zigzag(β^Φ). We can compose generalized graph morphisms by joining zigzags.

Definition 4 (Connection relation).
Let $\boldsymbol{\delta} = (\delta^\Phi,\ \delta : \delta^\Phi \to \mathcal{C}_0)$ be a diagram. Two arrows $u : A \to B$ and $v : A \to C$ of $\mathcal{C}_0$ are said to be *connected* by the diagram $\boldsymbol{\delta}$ if and only if there exist a zigzag on δ^Φ, $Z = n_0 \xrightarrow{a_0} n_1 \cdots n_{k-1} \xrightarrow{a_{k-1}} n_k$, and a set of arrows in $\mathcal{C}_0$, $\{c_i : A \to \delta(n_i);\ i \in \{0, \ldots, k\}\}$, such that:

- $u = c_0$ (and thus $\delta(n_0) = B$)
- $v = c_k$ (and thus $\delta(n_k) = C$)
- $\forall i \in \{0, \ldots, k-1\}$:
  $\delta(a_i) \circ c_i = c_{i+1}$, if a_i is oriented from n_i to n_{i+1};
  $\delta(a_i) \circ c_{i+1} = c_i$, if a_i is oriented from n_{i+1} to n_i.

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



Definition 5 (Category of finite diagrams DIAGR ($\mathcal{C}_0$)).
Let $\mathcal{C}_0$ be a category.

- An *object* $\boldsymbol{\delta}$ of DIAGR ($\mathcal{C}_0$) is a finite diagram.
- Let $\boldsymbol{\alpha} = (\alpha^\Phi,\ \alpha : \alpha^\Phi \to \mathcal{C}_0)$ and $\boldsymbol{\beta} = (\beta^\Phi,\ \beta : \beta^\Phi \to \mathcal{C}_0)$ be two diagrams. A *diagram morphism* from $\boldsymbol{\alpha}$ to $\boldsymbol{\beta}$ is a couple

$$\boldsymbol{\tau : \alpha \to \beta} = (\tau^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi,\ \tau : \alpha \overset{\bullet}{\rightsquigarrow} \beta \circ \tau^\Phi),$$

  where
  - $\tau^\Phi : \alpha^\Phi \rightsquigarrow \beta^\Phi$ is a generalized graph morphism.
  - $\tau : \alpha \overset{\bullet}{\rightsquigarrow} \beta \circ \tau^\Phi$ is a "generalized natural transformation" i.e. a set of arrows $\tau_n : \alpha(n) \to \beta(\tau^\Phi(n)),\ \forall n \in \text{Vertices}(\alpha^\Phi)$
    such that $\forall a : m \to n \in \text{Edges}(\alpha^\Phi),\ \ \tau_n \circ \alpha(a) \sim_\beta \tau_m\ \ [\tau^\Phi(a)]$

Note that if τ^{Φ} is a graph morphism, then τ is a natural transformation.

– Let us consider three diagrams and two diagram morphisms
$\boldsymbol{\alpha} = (\alpha^{\Phi},\ \alpha : \alpha^{\Phi} \to \mathcal{C}_0),\ \boldsymbol{\beta} = (\beta^{\Phi},\ \beta : \beta^{\Phi} \to \mathcal{C}_0),\ \boldsymbol{\gamma} = (\gamma^{\Phi},\ \gamma : \gamma^{\Phi} \to \mathcal{C}_0)$
$\boldsymbol{\sigma} : \boldsymbol{\alpha} \to \boldsymbol{\beta} = (\sigma^{\Phi} : \alpha^{\Phi} \rightsquigarrow \beta^{\Phi},\ \sigma : \alpha \overset{\bullet}{\rightsquigarrow} \beta \circ \sigma^{\Phi})$
$\boldsymbol{\tau} : \boldsymbol{\beta} \to \boldsymbol{\gamma} = (\tau^{\Phi} : \beta^{\Phi} \rightsquigarrow \gamma^{\Phi},\ \tau : \beta \overset{\bullet}{\rightsquigarrow} \gamma \circ \tau^{\Phi})$
The composition of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ is the couple
$$\tau \circ \sigma : \alpha \to \gamma = (\tau^{\Phi} \circ \sigma^{\Phi} : \alpha^{\Phi} \rightsquigarrow \gamma^{\Phi},\ \lambda : \alpha \overset{\bullet}{\rightsquigarrow} \gamma \circ \tau^{\Phi} \circ \sigma^{\Phi}),$$
where $\tau^{\Phi} \circ \sigma^{\Phi}$ is a composition of generalized graph morphisms, and λ is the "generalized natural transformation" defined by $\lambda_n = \tau_{\sigma^{\Phi}(n)} \circ \sigma_n$.

One can easily check that DIAGR $(\mathcal{C}_0)$ is indeed a category.

Colimits

The category $\mathcal{C}_0$ can be embedded in DIAGR $(\mathcal{C}_0)$ with a functor $I : \mathcal{C}_0 \to$ DIAGR $(\mathcal{C}_0)$. Colimits can be defined as usual [9]. In our setting, a cone from a diagram $\boldsymbol{\alpha}$ is a couple $(C,\ \boldsymbol{\lambda} : \boldsymbol{\alpha} \to I(C))$ where C is an object of $\mathcal{C}_0$ and $\boldsymbol{\lambda}$ is an arrow of DIAGR $(\mathcal{C}_0)$.

The cone $(C,\ \boldsymbol{\lambda} : \boldsymbol{\alpha} \to I(C))$ is a *colimiting cone* from $\boldsymbol{\alpha}$ if and only if for any cone $(D,\ \boldsymbol{\lambda'} : \boldsymbol{\alpha} \to I(D))$, there exists a unique arrow $\phi : C \to D$ such that $I(\phi) \circ \boldsymbol{\lambda} = \boldsymbol{\lambda'}$. C is called the *colimit* of $\boldsymbol{\alpha}$, and we note $C =$ Colim $\boldsymbol{\alpha}$. One diagram may have several colimits, which are then isomorphic. Writing $C =$ Colim $\boldsymbol{\alpha}$ means that we have *chosen* the object C for the colimit of $\boldsymbol{\alpha}$.

A category is finitely cocomplete when every finite diagram has a colimit.

Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be two diagrams with colimiting cones
(Colim $\boldsymbol{\alpha}$, $\boldsymbol{\eta_{\alpha}} : \boldsymbol{\alpha} \to I($Colim $\boldsymbol{\alpha}))$ and (Colim $\boldsymbol{\beta}$, $\boldsymbol{\eta_{\beta}} : \boldsymbol{\beta} \to I($Colim $\boldsymbol{\beta}))$.
Let $\boldsymbol{\sigma} : \boldsymbol{\alpha} \to \boldsymbol{\beta}$ be a diagram morphism. Then there exists a unique arrow Colim $\boldsymbol{\sigma}$: Colim $\boldsymbol{\alpha} \to$ Colim $\boldsymbol{\beta}$, such that $I($Colim $\boldsymbol{\sigma}) \circ \boldsymbol{\eta_{\alpha}} = \boldsymbol{\eta_{\beta}} \circ \boldsymbol{\sigma}$.

Theorem 6. *Let $\mathcal{C}_0$ be a finitely cocomplete category. Then*

1. *Colim : DIAGR $(\mathcal{C}_0) \to \mathcal{C}_0$ is a functor.*
2. *The mapping $\boldsymbol{\eta}$, which associates to each diagram $\boldsymbol{\alpha}$ the colimiting cone from $\boldsymbol{\alpha}$ to $I($Colim $\boldsymbol{\alpha})$ is a natural transformation $\boldsymbol{\eta} : Id_{\mathrm{DIAGR}(\mathcal{C}_0)} \overset{\bullet}{\to} I \circ$ Colim.*
3. *The functor Colim : DIAGR $(\mathcal{C}_0) \to \mathcal{C}_0$ is a left-adjoint for the functor $I : \mathcal{C}_0 \to$ DIAGR $(\mathcal{C}_0)$. The unit of the adjunction (Colim $\dashv I$) is the natural transformation $\boldsymbol{\eta}$.*

4.2 The Category diagr $(\mathcal{C}_0)$

In the category DIAGR $(\mathcal{C}_0)$, different arrows may have equal colimits. For instance, for defining the arrow $\boldsymbol{\sigma}$ from $\boldsymbol{\alpha}$ to $\boldsymbol{\delta}_1$ (Fig. 3), we can associate the vertex 1 either to the vertex $2'$, $4'$, $1'$, $5'$ or $3'$. (Of course association of edges to zigzags must be done accordingly). Those different arrows have the same colimit. The same way, non isomorphic objects may have isomorphic colimits in DIAGR $(\mathcal{C}_0)$. For instance, $\boldsymbol{\delta}_1 \ncong \boldsymbol{\delta}_3$, but Colim $\boldsymbol{\delta}_1 \cong$ Colim $\boldsymbol{\delta}_3$ (Fig. 2). The aim of this paragraph is to define a category where equalities of colimiting arrows and isomorphisms of colimit objects will be reflected at the level of diagrams.

Definition 7. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be two diagrams. Let $\boldsymbol{\sigma}, \boldsymbol{\tau} : \boldsymbol{\alpha} \to \boldsymbol{\beta}$ be two arrows of $\mathrm{DIAGR}\,(\mathcal{C}_0)$. We define the relation $\approx$ on arrows of $\mathrm{DIAGR}\,(\mathcal{C}_0)$ as follows:

$$\boldsymbol{\sigma} \approx \boldsymbol{\tau} \;\Leftrightarrow\; \forall n \in \mathrm{Vertices}(\alpha^{\Phi}) : \sigma_n \sim_{\beta} \tau_n$$

Theorem 8. *The relation $\approx$ is a congruence*

Definition 9. As $\approx$ is a congruence, we can consider the quotient category

$$\mathrm{diagr}\,(\mathcal{C}_0) = \mathrm{DIAGR}\,(\mathcal{C}_0)/\approx$$

Let $[-] : \mathrm{DIAGR}\,(\mathcal{C}_0) \to \mathrm{diagr}\,(\mathcal{C}_0)$ be the associated projection functor.

Theorem 10. *Let $\mathcal{C}_0$ be a finitely cocomplete category. The functor Colim is compatible with $\approx$. In other words, let $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ be two diagrams, and $\boldsymbol{\sigma}, \boldsymbol{\tau} : \boldsymbol{\alpha} \to \boldsymbol{\beta}$ two arrows of DIAGR $(\mathcal{C}_0)$. We have: $\boldsymbol{\sigma} \approx \boldsymbol{\tau} \Rightarrow Colim\,\boldsymbol{\sigma} = Colim\,\boldsymbol{\tau}$.*

The category $\mathrm{diagr}\,(\mathcal{C}_0)$ is finitely cocomplete. We have to show that every diagram over $\mathrm{diagr}\,(\mathcal{C}_0)$ — i.e. every object of $\mathrm{DIAGR}\,(\mathrm{diagr}\,(\mathcal{C}_0))$ — has a colimit in $\mathrm{diagr}\,(\mathcal{C}_0)$. We first define an operation of flattening

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

Intuitively, flattening the diagram of diagrams $\boldsymbol{\Delta}$ consists in considering the union of all subdiagrams of $\boldsymbol{\Delta}$, and in transforming every arrow of $\mathrm{DIAGR}\,(\mathcal{C}_0)$ into a set of arrows of $\mathcal{C}_0$.

Definition 11 (Flattening $\mu : \mathbf{DIAGR}\,(\mathbf{DIAGR}\,(\mathcal{C}_0)) \to \mathbf{DIAGR}\,(\mathcal{C}_0)$). Let $\boldsymbol{\Delta} = (\Delta^{\Phi}, \Delta : \Delta^{\Phi} \to \mathrm{DIAGR}\,(\mathcal{C}_0))$ be an object of $\mathrm{DIAGR}\,(\mathrm{DIAGR}\,(\mathcal{C}_0))$. We define the diagram $\mu(\boldsymbol{\Delta}) = \boldsymbol{\delta} = (\delta^{\Phi},\ \delta : \delta^{\Phi} \to \mathcal{C}_0)$ as follows:

- δ^{Φ} is a graph, given by $\mathrm{Vertices}(\delta^{\Phi})$ and $\mathrm{Edges}(\delta^{\Phi})$.

$$\mathrm{Vertices}(\delta^{\Phi}) = \{\ (N, n_N)\ ;\ N \in \mathrm{Vertices}(\Delta^{\Phi})\ ;\ \ n_N \in \mathrm{Vertices}(\Delta(N)^{\Phi})\ \}$$

$$\begin{aligned}\mathrm{Edges}(\delta^{\Phi}) \quad = \{\ & (N, a_N) : (N, n_N) \to (N, n'_N)\ ; \\ & N \in \mathrm{Vertices}(\Delta^{\Phi})\ ;\ \ n_N, n'_N \in \mathrm{Vertices}(\Delta(N)^{\Phi})\ ; \\ & a_N : n_N \to n'_N \in \mathrm{Edges}(\Delta(N)^{\Phi})\ \} \\ \cup\ \{\ & (A : N \to N', n_N) : (N, n_N) \to (N', \Delta(A)^{\Phi}(n_N))\ ; \\ & N, N' \in \mathrm{Vertices}(\Delta^{\Phi})\ ;\ \ A : N \to N' \in \mathrm{Edges}(\Delta^{\Phi})\ ; \\ & n_N \in \mathrm{Vertices}(\Delta(N)^{\Phi})\ \}\end{aligned}$$

- $\delta : \delta^{\Phi} \to \mathcal{C}_0$ is a functor, given by its action on vertices and edges of δ^{Φ}.
  - Action on vertices: $\delta(N, n_N)$ is an object of $\mathcal{C}_0$, isomorphic to $\Delta(N)(n_N)$. We call this isomorphism $(J_N)_{n_N} : \Delta(N)(n_N) \to \delta(N, n_N)$.
  - Action on edges: $\delta(N, a_N) = (J_N)_{n'_N} \circ \Delta(N)(a) \circ (J_N)^{-1}_{n_N}$
    $\delta(A, n_N) = (J_{N'})_{\Delta(A)^{\Phi}(n_N)} \circ \Delta(A)_{n_N} \circ (J_N)^{-1}_{n_N}$

Theorem 12. *The category diagr $(\mathcal{C}_0)$ is finitely cocomplete.*

Proof sketch. Let $\boldsymbol{\Gamma}$ be an object of $\mathrm{DIAGR}\,(\mathrm{diagr}\,(\mathcal{C}_0))$. We suppose we are able to choose a representative for each equivalence class of arrows, i.e. we have a graph morphism $Rep : \mathrm{diagr}\,(\mathcal{C}_0) \to \mathrm{DIAGR}\,(\mathcal{C}_0)$ which is the identity on objects. We define a diagram $\boldsymbol{\Delta}$ in $\mathrm{DIAGR}\,(\mathrm{DIAGR}\,(\mathcal{C}_0))$ as

$$\boldsymbol{\Delta} = Rep \circ \boldsymbol{\Gamma} = (\Gamma^{\Phi},\ Rep \circ \Gamma : \Gamma^{\Phi} \to \mathrm{DIAGR}\,(\mathcal{C}_0))$$

Let $\boldsymbol{\delta} = [\mu(\boldsymbol{\Delta})]$. Let $(\eta_{\Gamma})_N = [J_N]$. Then $(\boldsymbol{\delta}, \eta_{\Gamma} : \boldsymbol{\Gamma} \to I(\boldsymbol{\delta}))$ is a colimiting cone from $\boldsymbol{\Gamma}$.

As usual, colimits are defined up to isomorphisms. To define $\boldsymbol{\delta}$, we made two choices: μ is defined up to isomorphisms on objects of $\mathcal{C}_0$, and Rep contains a choice of arrow to represent an equivalence class.

Theorem 13. *Let $\mathcal{C}$ be a finitely cocomplete category. For every functor $F : \mathcal{C}_0 \to \mathcal{C}$ there exists a functor $H : diagr\,(\mathcal{C}_0) \to \mathcal{C}$, unique up to isomorphism, such that $H \circ [-] \circ I \cong F$ and for every diagram $\boldsymbol{\Gamma}$ of $DIAGR\,(diagr\,(\mathcal{C}_0))$, $Colim\,(H \circ \boldsymbol{\Gamma}) \cong H(Colim\,\boldsymbol{\Gamma})$.*

5 Representing Modular Specifications as Diagrams

We associate modular specifications to diagrams and specification morphisms to diagram morphisms in $\mathrm{diagr}\,(\mathcal{C}_0)$. As $\mathrm{diagr}\,(\mathcal{C}_0)$ is finitely cocomplete, with chosen colimits, there exists a unique homomorphism (which is also a functor)

$$\mathcal{D}[\![]\!] : \mathrm{Term}\,(\mathcal{C}_0) \to \mathrm{diagr}\,(\mathcal{C}_0)$$

such that $\mathcal{D}[\![]\!] \circ e = [-] \circ I$. $\mathcal{D}[\![]\!]$ associates to each categorical construction in $\mathrm{Term}\,(\mathcal{C}_0)$ the corresponding construction in $\mathrm{diagr}\,(\mathcal{C}_0)$:

$\mathcal{D}[\![Sp]\!] = [I(Sp)]$, if Sp is a specification of $\mathcal{C}_0$
$\mathcal{D}[\![p]\!] = [I(p)]$, if p is a specification morphism of $\mathcal{C}_0$
$\mathcal{D}[\![g \,;\, f]\!] = \mathcal{D}[\![g]\!] \circ \mathcal{D}[\![f]\!]$ (composition of arrows in $\mathrm{diagr}\,(\mathcal{C}_0)$)
$\mathcal{D}[\![\mathsf{id}_A]\!] = id_{\mathcal{D}[\![A]\!]}$ (identity arrow of $\mathrm{diagr}\,(\mathcal{C}_0)$)
$\mathcal{D}[\![\emptyset]\!] = \bigcirc$ ($\bigcirc$ is the empty diagram, initial object of $\mathrm{diagr}\,(\mathcal{C}_0)$)
$\mathcal{D}[\![\mathsf{j}_A]\!] = j_{\mathcal{D}[\![A]\!]}$ (unique arrow from the empty diagram to $\mathcal{D}[\![A]\!]$)

$$\mathcal{D}[\![\mathsf{push}\,(A,B,C,f,g)]\!] = push\,(\mathcal{D}[\![A]\!], \mathcal{D}[\![B]\!], \mathcal{D}[\![C]\!], \mathcal{D}[\![f]\!], \mathcal{D}[\![g]\!])$$
$$\mathcal{D}[\![\&_1(A,B,C,f,g)]\!] = \&_1(\mathcal{D}[\![A]\!], \mathcal{D}[\![B]\!], \mathcal{D}[\![C]\!], \mathcal{D}[\![f]\!], \mathcal{D}[\![g]\!])$$
$$\mathcal{D}[\![\&_2(A,B,C,f,g)]\!] = \&_2(\mathcal{D}[\![A]\!], \mathcal{D}[\![B]\!], \mathcal{D}[\![C]\!], \mathcal{D}[\![f]\!], \mathcal{D}[\![g]\!])$$
$$\mathcal{D}[\![\mathsf{up}\,(A,B,C,D,f,g,f',g')]\!] = up\,(\mathcal{D}[\![A]\!], \mathcal{D}[\![B]\!], \mathcal{D}[\![C]\!], \mathcal{D}[\![D]\!], \mathcal{D}[\![f]\!], \ldots, \mathcal{D}[\![g']\!])$$

From Theorem 13, there exists a functor $eval : \mathrm{diagr}\,(\mathcal{C}_0) \to \mathrm{SPEC}$ such that $eval \circ [-] \circ I \cong i$. This functor maps each diagram to the specification it represents.

Theorem 14. *There is a natural isomorphism $eval \circ \mathcal{D}[\![]\!] \cong \mathcal{S}[\![]\!]$*

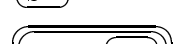
This theorem states that the calculus of diagrams is *sound*. The specification associated to a diagram coincides with the specification given by the standard semantics. More precisely, the colimit of a diagram representing a specification is isomorphic to this specification. The calculus of diagrams is also *complete*, in the sense that two isomorphic specifications in Term $(\mathcal{C}_0)$ correspond to isomorphic diagrams in diagr $(\mathcal{C}_0)$. This comes from the fact that $\mathcal{D}[\![]$ is well defined, because diagr $(\mathcal{C}_0)$ is finitely cocomplete.

Let us compute the diagram associated to the specification R_3 of section 3.

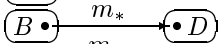
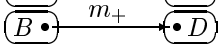
$P'_3 = \mathsf{push}\,(S, M, G, b \circ s, m \circ b \circ s)$
$R_3 = \mathsf{push}\,(\mathsf{push}\,(\emptyset, B, B, \mathsf{j}_B, \mathsf{j}_B), D, P'_3, \mathsf{up}\,(\emptyset, B, B, D, \mathsf{j}_B, \mathsf{j}_B, m_*, m_+),$
$\qquad \mathsf{up}\,(\emptyset, B, B, P'_3, \mathsf{j}_B, \mathsf{j}_B, \&_1(P'_3) \circ b, \&_2(P'_3) \circ m \circ b))$

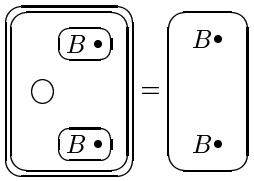
$\mathcal{D}[\![P'_3]\!] = push(\mathcal{D}[\![S]\!], \mathcal{D}[\![M]\!], \mathcal{D}[\![G]\!], \mathcal{D}[\![b]\!] \circ \mathcal{D}[\![s]\!], \mathcal{D}[\![m]\!] \circ \mathcal{D}[\![b]\!] \circ \mathcal{D}[\![s]\!])$

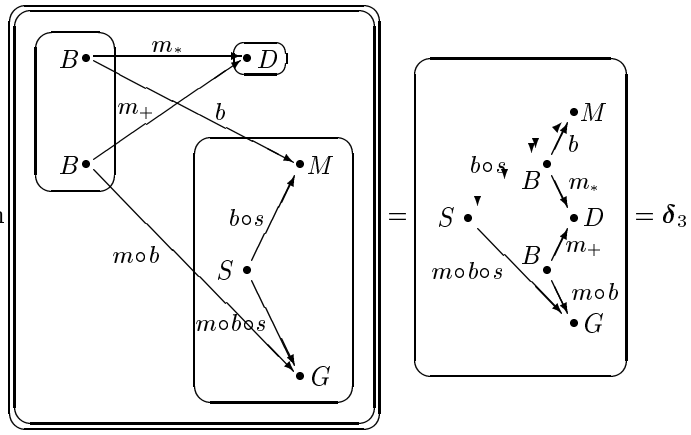
$\mathcal{D}[\![S]\!] = [I(S)] =$  $\quad \mathcal{D}[\![M]\!] =$  $\quad \mathcal{D}[\![G]\!] =$ 

$\mathcal{D}[\![b]\!] \circ \mathcal{D}[\![s]\!] =$  $\quad \mathcal{D}[\![m]\!] \circ \mathcal{D}[\![b]\!] \circ \mathcal{D}[\![s]\!] =$

$\mathcal{D}[\![P'_3]\!] = \mathrm{Colim}$

$\mathcal{D}[\![\emptyset]\!] = \bigcirc$
$\mathcal{D}[\![B]\!] =$ 
$\mathcal{D}[\![\mathsf{j}_B]\!] = \bigcirc \to$ 
$\mathcal{D}[\![D]\!] =$ 
$\mathcal{D}[\![m_*]\!] =$ 
$\mathcal{D}[\![m_+]\!] =$ 

$\mathcal{D}[\![\mathsf{push}\,(\emptyset, B, B, \mathsf{j}_B, \mathsf{j}_B)]\!] = \mathrm{Colim}$ 

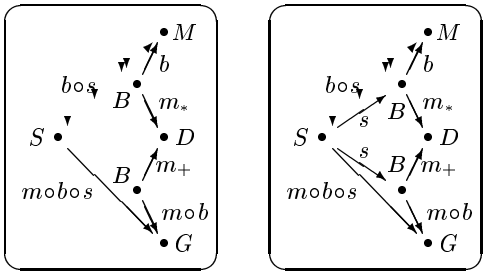
$\mathcal{D}[\![R_3]\!] = \mathrm{Colim}$  $= \boldsymbol{\delta}_3$

6 Isomorphisms of Diagrams

We have seen how to associate specifications to diagrams. Now the problem is to detect isomorphisms in diagr $(\mathcal{C}_0)$. In this section, we briefly describe an algorithm allowing to detect isomorphisms of diagrams in the restricted case when the base category $\mathcal{C}_0$ is finite and has no cycle. By "having no cycle" we mean here that any arrow from an object A to itself is the identity.

We make this restriction because we have not solved the problem in the general case. This restriction is compatible with the LPG language, because the definition of cycling morphisms is syntactically forbidden.

In order to simplify the presentation, we also suppose that there is no isomorphism between objects in the category $\mathcal{C}_0$. So in $\mathcal{C}_0$, $A \cong B \Rightarrow A = B$. The algorithm is actually not much more complicated without this restriction.



$\boldsymbol{\delta}_3$ $\boldsymbol{\delta}'_3$

Fig. 4. the completion of diagram $\boldsymbol{\delta}_3$ is $\boldsymbol{\delta}'_3$

The algorithm can be described in 3 steps:

1 Complete the diagram
For each pair of arrows $f : A \to B$ and $g : C \to B$, if there exists an arrow $h : A \to C$ such that $g \circ h = f$ in $\mathcal{C}_0$, we add the arrow h to the diagram. If h is the identity function id_A, then we merge both vertices labeled by A.
For instance completing the diagram $\boldsymbol{\delta}_3$ gives the diagram $\boldsymbol{\delta}'_3$ (Fig. 4). The diagrams $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ and $\boldsymbol{\alpha}$ are already complete.

2 Match the "terminating vertices"
A "terminating vertex" is a vertex where no edge starts from.

Lemma 15. *If two diagrams $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are complete and isomorphic, then for every terminating vertex m of $\boldsymbol{\alpha}$, there exists a terminating vertex n of $\boldsymbol{\beta}$ such that $\alpha(m) \cong \beta(n)$.*

Proof. use the fact that the diagrams are complete, and $\mathcal{C}_0$ has no cycle.

We can check that the diagrams $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$, and $\boldsymbol{\delta}'_3$ have three terminating vertices, labeled by M, D and G.

3 Match the "elementary zigzags"

An elementary zigzag of a diagram $\boldsymbol{\alpha}$ is a zigzag $n' \xleftarrow{f} n \xrightarrow{g} n''$ of $\boldsymbol{\alpha}$, where f and g may be compositions of arrows of $\mathcal{C}_0$, and $f \neq g$.

Definition 16 (Ordering on elementary zigzags). Let $\boldsymbol{\alpha}$ be a complete diagram. We define an ordering on elementary zigzags of $\boldsymbol{\alpha}$ as follows.
Let $Z = n'_0 \xleftarrow{u} n \xrightarrow{v} n'_k$ be an elementary zigzag of $\boldsymbol{\alpha}$. Let Z' be a zigzag of $\boldsymbol{\alpha}$, composed of the elementary zigzags $Z_1, ...Z_k$, with $Z'_i = n'_{i-1} \longleftarrow n_i \longrightarrow n'_i$.

- $Z < Z'_i$ iff $(u \sim_\alpha v\ [Z']$ and $\forall i, n \neq n_i)$.
- $Z \leq Z'_i$ iff $(Z = Z'$ or $Z < Z')$

To prove that $\leq$ is indeed an ordering, one has to use the fact that $\mathcal{C}_0$ has no cycle, and that the vertices linked by an identity arrow have been merged in the diagram $\boldsymbol{\alpha}$.

For instance, in $\boldsymbol{\delta}'_3$, (and in $\boldsymbol{\delta}_2$ as well)

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

$\leq$ is an ordering on a finite set, so there are maximal elements (maximal elementary zigzags). Intuitively, the maximal elementary zigzags are those which really count. The others can be removed without changing the colimit of the diagram.

Lemma 17. *If two diagrams $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are complete and isomorphic, then there exists an isomorphism from $\boldsymbol{\alpha}$ to $\boldsymbol{\beta}$ which associates every maximal elementary zigzag of $\boldsymbol{\alpha}$ to a maximal elementary zigzag of $\boldsymbol{\beta}$.*

Finally, two diagrams are isomorphic if they have the same terminating vertices, linked by the same maximal elementary zigzags. In particular, $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$ and $\boldsymbol{\delta}'_3$ are isomorphic.

7 Conclusion

We proposed to study "construction isomorphisms" of modular specifications. A "construction isomorphism" is an isomorphism which comes from general properties of colimits. We think this isomorphism is interesting, because it relies on the construction of a modular specification, without depending on the specific steps chosen for the construction. We showed in this paper that these isomorphisms of modular specifications correspond to isomorphisms of diagrams in the category $\mathrm{diagr}\,(\mathcal{C}_0)$. The category $\mathrm{diagr}\,(\mathcal{C}_0)$ is a completion with finite colimits of the category $\mathcal{C}_0$ of base specifications. In particular, $\mathrm{diagr}\,(\mathcal{C}_0)$ is finitely cocomplete, even if $\mathcal{C}_0$ is not finitely cocomplete. We showed how specifications can be associated to diagrams. We gave an algorithm to detect isomorphisms in $\mathrm{diagr}\,(\mathcal{C}_0)$, which strongly relies on the assumption that $\mathcal{C}_0$ has no cycle. We have not solved the problem in the general case, i.e. if there are cycles in $\mathcal{C}_0$

(which may introduce cycles in the diagram while completing it). We think this problem is much more difficult, because the number of arrows to consider may be infinite. However, in the case of algebraic specification, arrows are specification morphisms between finite signatures, so the number of arrows between two specifications remains finite, which suggests that isomorphisms should still be detectable.

References

1. D. Bert and R. Echahed. Design and implementation of a generic, logic and functional programming language. In *Proceedings of ESOP'86*, number 213 in LNCS, pages 119–132. Springer-Verlag, 1986.
2. M. Bidoit. The stratified loose approach: A generalization of initial and loose semantics. Technical Report 402, Université d'Orsay, France, 1988.
3. R.M. Burstall and J.A. Goguen. Putting theories together to make specifications. In *Int. Conf. Artificial Intelligence*, 1977.
4. R.M. Burstall and J.A. Goguen. The semantics of CLEAR, a specification language. In *Proc. Advanced Course on Abstract Software Specification*, number 86 in LNCS, pages 292–332. Springer-Verlag, 1980.
5. J. Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.
6. 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.
7. K. Futatsugi, J.A. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proc. Principles of Programming Languages*, pages 52–66, 1985.
8. M.-C. Gaudel. A first introduction to PLUSS. Technical report, Université d'Orsay, France, 1984.
9. S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.
10. J.-C. Reynaud. Putting algebraic components together: A dependent type approach. Research Report 810 I IMAG, LIFIA, Apr. 1990.
11. J.-C. Reynaud. Putting algebraic components together: A dependent type approach. Number 429 in LNCS. Springer-Verlag, 1990.
12. J.-C. Reynaud. Isomorphism of typed algebraic specifications. Internal Report, LGI-IMAG, Feb. 1993.
13. 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.
14. M. Wirsing. Structured Algebraic Specifications: A Kernel Language. *Theoretical Computer Science*, 42:123–249, 1986.

This article was processed using the LaTeX macro package with LLNCS style