Towards Diagrammatic Specifications of Symbolic Computation Systems*

C. Domínguez[1], D. Duval[2], L. Lambán[1], and J. Rubio[1]

[1] Departamento de Matemáticas y Computación, Universidad de La Rioja, Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño, La Rioja, Spain.
{cedomin,lalamban,jurubio}@dmc.unirioja.es
[2] Laboratoire de Modélisation et Calcul LMC-IMAG, Université Joseph Fourier, B.P. 53, 38041 Grenoble Cedex 9, France.
Dominique.Duval@imag.fr

Abstract. The aim of this work is to present an ongoing project to formalize, in the framework of diagrammatic logic (due to Dominique Duval and Christian Lair) some data structures appearing in Sergeraert's symbolic computation systems Kenzo and EAT. More precisely, we intend to translate into the diagrammatic setting a previous work based on standard algebraic specification techniques. In particular, we give hints on the reason why an important construction (called *imp* construction) in the specification of the systems can be understood as a freely generating functor between suitable categories of diagrammatic realizations. Even if very partial, these positive results seem to indicate that this new kind of specification is promising in the field of symbolic computation.

1 Introduction

Kenzo [7] and its predecessor EAT [14] are software systems developed by F. Sergeraert. They are devoted to Symbolic Computation in Algebraic Topology. Particularly, they carry out calculations of homology groups of complex topological spaces, namely iterated loop spaces. By means of EAT and Kenzo, some homology groups that had never been obtained with any other method, neither theoretical nor automatic, have been computed. In view of the obtained results, some years ago, three of the authors of this paper began the formal study of the programs, in order to reach a good understanding on the internal calculation processes of these software systems.

In particular, first we studied the data types used in EAT and Kenzo. In that study [10, 9, 5, 6] we found that there are two different *layers* of data structures in the systems. In the first layer, one finds the usual data structures. For instance, the system handles integer numbers, (finite) lists or trees of symbols (to represent linear combinations or polynomials) and so on. In the second layer, one must deal with algebraic structures like (graded) groups, rings, simplicial sets or chain complexes, whose elements are data that belong to the first layer.

* Partially supported by SEUI-MEC, project TIC2002-01626

Dagstuhl Seminar Proceedings 05021
Mathematics, Algorithms, Proofs
http://drops.dagstuhl.de/opus/volltexte/2006/292

We first realized that in a system such as EAT we are not only implementing an Abstract Data Type, or, shortly, an ADT (as a group, for instance), but also dealing with implementations of ADTs (several hundreds of *implementations* of the ADT group would populate the program memory). In [10] an operation, which is called *imp* construction, is defined. This construction models the step from a kind of structures to families of these structures. Besides, working with implementations in [10] we were able to prove that EAT (second-layer) data structures are as general as possible, in the sense that they are ingredients of final objects in certain categories of ADT implementations. Later on, led by this characterization of EAT data structures, in [9] we reinterpreted our results in terms of object-oriented technologies like hidden algebras or coalgebras.

In this paper we extend our interpretation of this construction and concepts using diagrammatic specifications developed by Duval and Lair [8]. This construction can be understood as a freely generating functor between categories of diagrammatic realizations.

The paper is organized as follows. In the section below some definitions and examples on diagrammatic specifications are introduced. In Section 3 the *imp* construction is briefly presented. Section 4 is devoted to explain a diagrammatic interpretation of the *imp* construction. The paper ends with a section of conclusions and future work.

2 Diagrammatic Logic

Diagrammatic specifications were introduced in [8], as summarized below. The basic tool for building diagrammatic specifications is the theory of *sketches* [4, 2, 3]. On the one hand, diagrammatic specifications generalize sketches, and on the other hand, they are defined by means of *projective sketches* at a meta-level: a diagrammatic specification *has* models, and at the same time it *is* a realization (this word is used at the meta-level instead of model) of a projective sketch.

Definition 1. *A* (directed) graph *is made of a set of points, a set of arrows and two maps from arrows to points. These maps assign to each arrow its* source *and* target*. An arrow g with source G_1 and target G_2 will be denoted by $g\colon G_1 \longrightarrow G_2$. A* morphism of graphs $\gamma\colon \mathcal{G} \longrightarrow \mathcal{G}'$ *consists of two maps, both denoted by γ, from the points of $\mathcal{G}$ (the arrows of $\mathcal{G}$, respectively) to the points of $\mathcal{G}'$ (to the arrows of $\mathcal{G}'$, respectively) such that, for each arrow $g\colon G_1 \longrightarrow G_2$ of $\mathcal{G}$, the source of $\gamma(g)$ is $\gamma(G_1)$ and its target is $\gamma(G_2)$.*

Definition 2. *A* compositive graph $\mathcal{G}$ *is made of a directed graph, called the* support *of $\mathcal{G}$, together with:*
– for some points A, a distinguished arrow $\mathrm{id}_A\colon A \longrightarrow A$ *which is called the* identity *at A,*
– for some consecutive pairs of arrows $(f\colon G_1 \longrightarrow G_2, g\colon G_2 \longrightarrow G_3)$ an arrow $g \circ f\colon G_1 \longrightarrow G_3$. This arrow is called the composite *of f and g.*
A morphism of compositive graphs *is a morphism of directed graphs which preserves identity arrows and composites.*

Definition 3. *A* cone *in a compositive graph $\mathcal{G}$ consists of a point V of $\mathcal{G}$ (the* vertex *of the cone), a morphism $b\colon \mathcal{I} \longrightarrow \mathcal{G}$ (the* base *of the cone) where $\mathcal{I}$ is a compositive graph and, for each point I in $\mathcal{I}$, an arrow $p_I\colon V \longrightarrow b(I)$ (the* projections *of the cone) such that, for each arrow $i\colon I \longrightarrow I'$ in $\mathcal{I}$, the composite $b(i) \circ p_I$ exists and $b(i) \circ p_I = p_{I'}$.*

Definition 4. *A* projective sketch *is a compositive graph $\mathcal{G}$ together with a set of cones in $\mathcal{G}$. These cones are called* distinguished cones.
A morphism of projective sketches *is a morphism of compositive graphs which preserves the distinguished cones.*

Note 1. It is clear that the notion of compositive graph is weaker than (the notion of) *category*. Recall that a category is a compositive graph such that each point has an identity arrow and each pair of consecutive arrows has a composite, moreover, the associativity and unitarity axioms hold:
– if (f, g) and (g, h) are consecutive, then $(h \circ g) \circ f = h \circ (g \circ f)$,
– if $f\colon X \longrightarrow Y$, then $f \circ \mathrm{id}_X = f$ and $\mathrm{id}_Y \circ f = f$.
A *morphism of categories*, or *functor*, is a morphism of compositive graphs.

In a category, *limits* are cones with a universal property. For instance, a *product* is a limit. The product of the points $Y_1, \ldots, Y_n$ is given by a cone:

$$\begin{array}{ccccc} & & P & & \\ & {\scriptstyle p_1}\swarrow & & \searrow{\scriptstyle p_n} & \\ Y_1 & & \cdots & & Y_n \end{array}$$

In this case, the universal property of the product establishes that for each cone with the same base:

$$\begin{array}{ccccc} & & X & & \\ & {\scriptstyle f_1}\swarrow & & \searrow{\scriptstyle f_n} & \\ Y_1 & & \cdots & & Y_n \end{array}$$

there is a unique arrow $(f_1, \ldots, f_n)\colon X \longrightarrow P$, called *factorization* of $f_1, \ldots, f_n$, such that $p_i \circ (f_1, \ldots, f_n) = f_i$ for each $i \in \{1, \ldots, n\}$.

When $n = 0$, this property says that, for each point X, there is a unique arrow from X to P; this means that P is a *terminal point* in the category.

Here is another example. If we impose that the following diagram is a limit:

$$\begin{array}{ccccc} & & P & & \\ & {\scriptstyle id_P}\swarrow & \downarrow & \searrow{\scriptstyle id_P} & \\ P & & {\scriptstyle m} & & P \\ & {\scriptstyle m}\searrow & \downarrow & \swarrow{\scriptstyle m} & \\ & & Q & & \end{array}$$

then the universal factorization property implies that m is a monomorphism (that is, for each pair of arrows f_l and f_r, if $m \circ f_l = m \circ f_r$ then $f_l = f_r$). In that case m is represented as $m\colon P \rightarrowtail Q$.

For instance, in the category of sets:

– A product is a cartesian product, which means that $P = Y_1 \times \ldots \times Y_n$, and $p_1, \ldots, p_n$ are the projections. The factorization arrow $(f_1, \ldots, f_n)\colon X \longrightarrow Y_1 \times \ldots \times Y_n$ builds n-uples of images.
– When $n = 0$, P is a singleton, i.e., a one-element set;
– A monomorphism is an injective map.

Definition 5. *A* (set-valued) realization S *of a projective sketch* $\mathcal{E}$ *maps each point* E *of* $\mathcal{E}$ *to a set* $S(E)$ *and each arrow* $e\colon E \longrightarrow E'$ *of* $\mathcal{E}$ *to a map* $S(e)\colon S(E) \longrightarrow S(E')$*, in such a way that each identity arrow becomes an identity map, each composite arrow becomes the corresponding composite map, and each distinguished cone becomes a limit.*
A morphism $\sigma\colon S_1 \longrightarrow S_2$ of realizations *between two realizations* S_1 *and* S_2 *of* $\mathcal{E}$ *is a natural transformation. That is, for each point* E *of* $\mathcal{E}$ *a map* $\sigma_E\colon S_1(E) \longrightarrow S_2(E)$ *in such a way that* $S_2(e) \circ \sigma_E = \sigma_{E'} \circ S_1(e)$ *for each arrow* $e\colon E \longrightarrow E'$ *of* $\mathcal{E}$*.*
The category of realizations of a projective sketch $\mathcal{E}$ *is denoted by* $\mathcal{R}eal(\mathcal{E})$*.*

Example 1. Let $\mathcal{E}_{Gr}$ be the projective sketch with two points and two arrows (and without any distinguished cone):

$$\mathcal{E}_{Gr} \ : \qquad \text{Type} \underset{\text{type}}{\overset{\text{context}}{\rightleftarrows}} \text{Term}$$

The idea is that the points Type and Term can be understood respectively as "points" and "arrows", and the arrows context and type as "source" and "target". Then, the category of realizations of $\mathcal{E}_{Gr}$ is the category of directed graphs.

Example 2. The projective sketch $\mathcal{E}_{Gr}$ can be enriched in order to get a projective sketch $\mathcal{E}_{Comp}$ such that the realizations of $\mathcal{E}_{Comp}$ are compositive graphs.

First, a point Cons is added to $\mathcal{E}_{Gr}$, for "pairs of consecutive arrows". In order to ensure that it will be interpreted as the set of pairs of consecutive arrows, it is accompanied by two arrows $\text{first}\colon \text{Cons} \longrightarrow \text{Term}$ and $\text{second}\colon \text{Cons} \longrightarrow \text{Term}$ (first and second arrow in the pair) and by the following distinguished cone with vertex Cons (as usual, the diagonal arrow $\text{context} \circ \text{second} = \text{type} \circ \text{first}\colon \text{Cons} \longrightarrow \text{Type}$ is omitted):

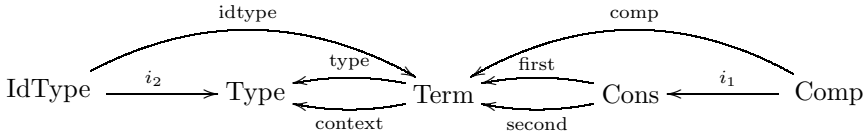
$$\begin{array}{ccccc} & & \text{Cons} & & \\ & \overset{\text{first}}{\swarrow} & & \overset{\text{second}}{\searrow} & \\ \text{Term} & & & & \text{Term} \\ & \underset{\text{type}}{\searrow} & & \underset{\text{context}}{\swarrow} & \\ & & \text{Type} & & \end{array}$$

Then, a new point Comp (for *composable pairs of terms*), a monomorphism $i_1\colon \text{Comp} \rightarrowtail \text{Cons}$ for the inclusion, and an arrow $\text{comp}\colon \text{Cons} \longrightarrow \text{Term}$ to represent the composite of composable terms, are added. Moreover, it is necessary to add the composites:

$$\text{context} \circ \text{comp} = \text{context} \circ \text{first} \ , \ \ \text{type} \circ \text{comp} = \text{type} \circ \text{second} \ .$$

In a similar way a point IdType for *types with identity*, a monomorphism $i_2\colon \mathrm{IdType} \rightarrowtail \mathrm{Type}$ for the inclusion, and an arrow $\mathrm{idtype}\colon \mathrm{IdType} \longrightarrow \mathrm{Term}$ for the *selection of the identity* of the type, are added. It is also necessary to add the composites:

$$\mathrm{context} \circ \mathrm{idtype} = i_2 \ , \ \mathrm{type} \circ \mathrm{idtype} = i_2 \ .$$

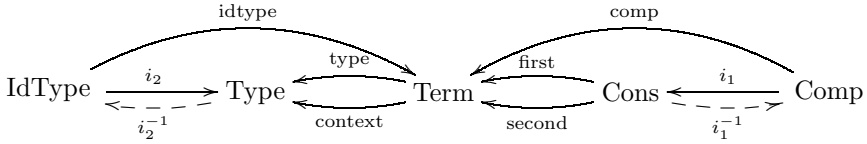


This enrichment allows to define a morphism of projective sketches $P_{Comp}\colon \mathcal{E}_{Gr} \longrightarrow \mathcal{E}_{Comp}$.

Example 3. We will say that a compositive graph is *saturated* if each pair of "composable terms" has a composite and each type has an identity. A sketch $\mathcal{E}_{SComp}$ of saturated compositive graphs is built by enriching $\mathcal{E}_{Comp}$ with an inverse for the arrows $i_1\colon \mathrm{Comp} \rightarrowtail \mathrm{Cons}$ and $i_2\colon \mathrm{IdType} \rightarrowtail \mathrm{Type}$. This means that two arrows $i_1^{-1}\colon \mathrm{Cons} \longrightarrow \mathrm{Comp}$ and $i_2^{-1}\colon \mathrm{Type} \longrightarrow \mathrm{IdType}$ are added, such that:

$$i_1 \circ i_1^{-1} = \mathrm{id}_{\mathrm{Cons}} \ , \ i_1^{-1} \circ i_1 = \mathrm{id}_{\mathrm{Comp}} \quad \text{and} \quad i_2 \circ i_2^{-1} = \mathrm{id}_{\mathrm{Type}} \ , \ i_2^{-1} \circ i_2 = \mathrm{id}_{\mathrm{IdType}} \ .$$

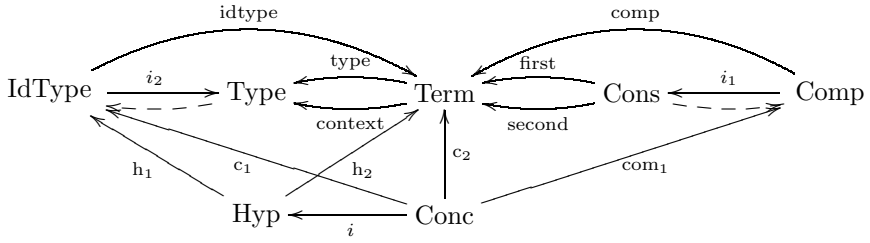
Then, the arrow $\mathrm{comp} \circ i_1^{-1}\colon \mathrm{Cons} \longrightarrow \mathrm{Term}$ stands for the *composition* of consecutive terms, and the arrow $\mathrm{idtype} \circ i_2^{-1}\colon \mathrm{Type} \longrightarrow \mathrm{Term}$ stands for the *selection of identity*:



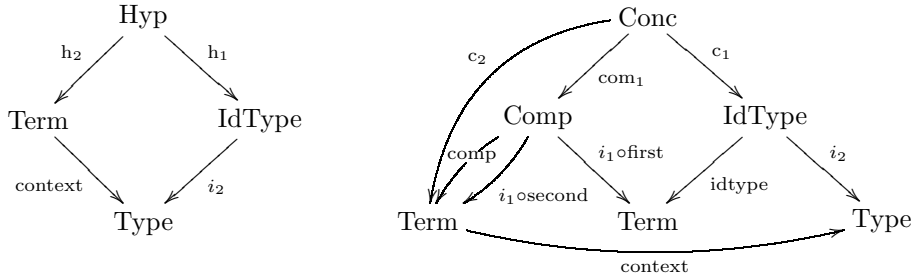
This enrichment defines a morphism of projective sketches $P_{SComp}\colon \mathcal{E}_{Comp} \longrightarrow \mathcal{E}_{SComp}$.

Example 4. Now, the projective sketch $\mathcal{E}_{SComp}$ can be enriched in order to define a projective sketch $\mathcal{E}_{Cat}$ for categories. To this aim, it is necessary to include the associativity and unitarity axioms in the projective sketch which has been just defined for saturated compositive graphs. These axioms can been seen as properties (or rules) with a hypothesis and a conclusion (which are required to be equivalent). For instance, the first part of the unitarity axiom can be seen as a property such that the hypothesis is made of a term $f\colon X \longrightarrow Y$ and the identity term at X, $id_X\colon X \longrightarrow X$, and the conclusion is made of these terms

f and id_X, and the composite $f \circ \mathrm{id}_X$ which must be equal to f. This rule can be imposed by enriching the projective sketch $\mathcal{E}_{SComp}$ with two new points Hyp and Conc in the following way:



Besides, this projective sketch also contains two distinguished cones with vertexes Hyp and Conc which determine a hypothesis and a conclusion:



Now, it must be observed that a conclusion contains a hypothesis. This is imposed with the arrow i and the composites $\mathrm{h}_1 \circ i = \mathrm{c}_1$ and $\mathrm{h}_2 \circ i = \mathrm{c}_2$. Finally, the rule is declared by enriching this sketch with an inverse arrow for i, that is $i^{-1}\colon \mathrm{Conc} \longrightarrow \mathrm{Hyp}$ such that:

$$i \circ i^{-1} = \mathrm{id}_{\mathrm{Hyp}} \quad \text{and} \quad i^{-1} \circ i = \mathrm{id}_{\mathrm{Conc}} \,.$$

This rule means that when the hypothesis is true, then the conclusion is also true.

Similar processes are needed in order to obtain the other unitarity axiom and the associativity axiom.

This enrichment defines a sketch morphism $P_{Cat}\colon \mathcal{E}_{SComp} \longrightarrow \mathcal{E}_{Cat}$.

2.1 The Adjunction Associated to a Sketch Morphism

Let us consider a sketch morphism:

$$P\colon \mathcal{E} \longrightarrow \overline{\mathcal{E}} \,.$$

In a natural way, we can define the *omitting functor* $\mathrm{G}_P\colon \mathcal{R}eal(\overline{\mathcal{E}}) \longrightarrow \mathcal{R}eal(\mathcal{E})$. It is given by $\mathrm{G}_P(\mathcal{C}) = \mathcal{C} \circ P$, for any realization $\mathcal{C}$ of $\overline{\mathcal{E}}$.

An important fact is that the omitting functor has a left adjoint $\mathrm{F}_P\colon \mathcal{R}eal(\mathcal{E}) \longrightarrow \mathcal{R}eal(\overline{\mathcal{E}})$. The functor F_P is called the *freely generating functor*.

In this way, if S is a realization of $\mathcal{E}$ and $\mathcal{C}$ is a realization of $\overline{\mathcal{E}}$ there exists a bijection:

$$\mathrm{Hom}_{\mathcal{R}eal(\mathcal{E})}(S, \mathrm{G}_P(\mathcal{C})) \cong \mathrm{Hom}_{\mathcal{R}eal(\overline{\mathcal{E}})}(\mathrm{F}_P(S), \mathcal{C}) \ .$$

$$\mathcal{R}eal(\mathcal{E}) \underset{\mathrm{G}_P}{\overset{\mathrm{F}_P}{\rightleftarrows}} \mathcal{R}eal(\overline{\mathcal{E}})$$

The existence of this left adjoint is a major property of projective sketches [4, 13].

Example 5. Let $P = P_{SComp} \circ P_{Comp} \colon \mathcal{E}_{Gr} \longrightarrow \mathcal{E}_{SComp}$ be the sketch morphism obtained by composition of two inclusions (graphs into compositive graphs, and compositive graphs into saturated graphs). The omitting functor G_P maps each saturated composite graph to its underlying graph.

The freely generating functor F_P maps each graph to its freely generated saturated compositive graph, which is obtained by adding all the missing identities and composites.

Definition 6. *A* propagator *is a morphism of projective sketches $P \colon \mathcal{E} \longrightarrow \overline{\mathcal{E}}$ such that the functor G_P is full and faithful.*

The following theorem shows a characterization for propagators [8].

Theorem 1. *A morphism of projective sketches is a propagator if and only if, up to equivalence, it consists of adding inverses to arrows.*

Example 6. The inclusion $P_{SComp} \colon \mathcal{E}_{Comp} \longrightarrow \mathcal{E}_{SComp}$ is a propagator. The inclusion of $\mathcal{E}_{Gr}$ in $\mathcal{E}_{SComp}$ from Example 5 is not a propagator.

2.2 Diagrammatic Specifications and Domains

Projective sketches and propagators can be used at the *meta-level* to define diagrammatic specifications. From now on, let us consider a propagator:

$$P \colon \mathcal{E} \longrightarrow \overline{\mathcal{E}} \ .$$

Definition 7. *A* (diagrammatic) P-specification *is a realization of $\mathcal{E}$, and a* (diagrammatic) P-domain *is a realization of $\overline{\mathcal{E}}$. That is,*

$$\mathcal{S}pec(P) = \mathcal{R}eal(\mathcal{E}) \quad , \quad \mathcal{D}om(P) = \mathcal{R}eal(\overline{\mathcal{E}}) \ .$$

So, the adjunction is now represented by:

$$\mathcal{S}pec(P) \underset{\mathrm{G}_P}{\overset{\mathrm{F}_P}{\rightleftarrows}} \mathcal{D}om(P)$$

Fixed an specification and a domain for a propagator, we define the notion of model for the specification with values in the domain.

Definition 8. *Let S denote a P-specification and $\mathcal{C}$ a P-domain. The* set of P-models of S with values in $\mathcal{C}$ *is:*

$$\mathrm{Mod}_P(S,\mathcal{C}) = \mathrm{Hom}_{\mathcal{D}om(P)}(\mathrm{F}_P(S),\mathcal{C}) \ .$$

The adjunction property yields the following bijection:

$$\mathrm{Mod}_P(S,\mathcal{C}) \cong \mathrm{Hom}_{\mathcal{S}pec(P)}(S,\mathrm{G}_P(\mathcal{C})) \ .$$

Moreover, for each morphism $\sigma\colon S \longrightarrow S'$, it is easy to define a map:

$$\mathrm{Mod}_P(\sigma,\mathcal{C}) : \mathrm{Mod}_P(S',\mathcal{C}) \longrightarrow \mathrm{Mod}_P(S,\mathcal{C}) \ .$$

In this way, we can establish that $\mathrm{Mod}_P(-,\mathcal{C})$ is a contravariant functor from the category of P-specifications to the category of sets.

In addition, in the examples we are going to develop, there will exist a natural notion of morphisms of P-models, and then, a *category of P-models of a specification with values in a domain.*

Example 7. Let $P = P_{SComp}$. A model M of a graph $\mathcal{G}$ with values in the category of sets interprets each point G of $\mathcal{G}$ as a set $M(G)$ and each arrow $g\colon G_1 \longrightarrow G_2$ of $\mathcal{G}$ as a map $M(g)\colon M(G_1) \longrightarrow M(G_2)$.

2.3 The Yoneda Morphism

The Yoneda morphism is a very classical construction in Category Theory [1] and plays a basic role in our diagrammatic framework. In the case of projective sketches, the Yoneda morphism will allow us to relate a sketch with the category of its realizations [13]. Basically, when $\mathcal{E}$ has as underlying compositive graph a category, its Yoneda morphism maps each point E of $\mathcal{E}$ to the realization that consists of the sets of arrows with source E in $\mathcal{E}$. In the other case, that is $\mathcal{E}$ is not a category, it is easy to build a sketch with this characteristic from $\mathcal{E}$ (by adding all the missing identities and composites, and by identifying arrows when it is needed for the unitarity and associativity axioms).

This Yoneda morphism is very useful in this framework. On the one hand, it is a natural way to "illustrate" things defined in a meta specification level using a specification level. On the other hand, it will give us a method for building the image of the freely generating functor associated to a morphism of projective sketches. Before introducing the definition of the Yoneda morphism the notion of *projective prototype* is presented.

Definition 9. *A* projective prototype *is a projective sketch such that its underlying compositive graph is a category and all its distinguished cones are limits.*

Theorem 2. *Each projective sketch $\mathcal{E}$ freely generates a projective prototype* $\mathrm{Proto}(\mathcal{E})$.

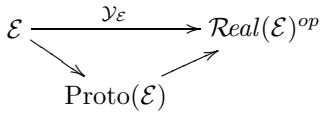
Let $\mathcal{E}$ be a projective sketch. Whenever $\mathcal{E}$ is a projective prototype (hence, a category), we define $\mathcal{Y}_{\mathcal{E}}$ as the contravariant functor given by: for any point E of $\mathcal{E}$, $\mathcal{Y}_{\mathcal{E}}(E) = \mathrm{Hom}_{\mathcal{E}}(E, -)$, and for any arrow $e\colon E \longrightarrow E'$ of $\mathcal{E}$,

$$\mathcal{Y}_{\mathcal{E}}(e) = \mathrm{Hom}_{\mathcal{E}}(e, -)\colon \mathcal{Y}_{\mathcal{E}}(E') \longrightarrow \mathcal{Y}_{\mathcal{E}}(E)$$

which maps each arrow $f\colon E' \longrightarrow E''$ to $f \circ e\colon E \longrightarrow E''$.

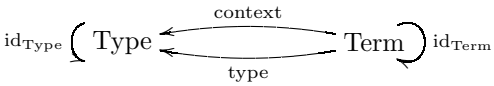
When $\mathcal{E}$ is any projective sketch, it freely generates a projective prototype $\mathrm{Proto}(\mathcal{E})$. So we define the Yoneda morphism $\mathcal{Y}_{\mathcal{E}}$ by the composition of the canonical morphism from $\mathcal{E}$ to $\mathrm{Proto}(\mathcal{E})$, followed by $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E})}$.

Definition 10. *The* Yoneda morphism *of $\mathcal{E}$ is the (contravariant) morphism:*



where $\mathcal{R}eal(\mathcal{E})^{op}$ is the opposite *category of $\mathcal{R}eal(\mathcal{E})$.*

Example 8. Let us consider the projective sketch $\mathcal{E}_{Gr}$ of graphs described in Example 1. Then, the prototype $\mathrm{Proto}(\mathcal{E}_{Gr})$ is simply the category:



In $\mathrm{Proto}(\mathcal{E}_{Gr})$, the identity arrow $\mathrm{id}_{\mathrm{Type}}$ is the unique arrow from Type to Type, and there is no arrow from Type to Term. So, it follows that the graph $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Gr})}(\mathrm{Type})$ has a unique point and has no arrows. So, a representation of this graph is:

Similarly, in $\mathrm{Proto}(\mathcal{E}_{Gr})$, the identity arrow $\mathrm{id}_{\mathrm{Term}}$ is the unique arrow from Term to Term, and there are two arrows from Term to Type. So, it follows that the graph $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Gr})}(\mathrm{Term})$ assigns as set for the point Term a set with one element f, and for the point Type a set with two elements Y, Z; which can be thought that are the context and the type of the unique arrow. So, a representation of this graph is:

In this way, a representation for the points Type and Term, which are defined in a meta specification level, is obtained in a specification level.

The following theorem states some of the properties of the Yoneda morphism [8].

Let $\mathcal{E}$ be a projective sketch and let S be a realization of $\mathcal{E}$. We consider the compositive graph $\mathcal{E} \setminus S$ given by: a point E_x for each point E of $\mathcal{E}$ and each $x \in S(E)$, an arrow $e_x \colon E_x \longrightarrow E'_{S(e)(x)}$ for each arrow $e \colon E \longrightarrow E'$ of $\mathcal{E}$ and each $x \in S(E)$. Moreover, $\mathcal{E} \setminus S$ has the identities $\mathrm{id}_{E_x} = (\mathrm{id}_E)_x$ when id_E exists in $\mathcal{E}$, and the composites $(e' \circ e)_x = e'_{S(e)(x)} \circ e_x$, when $e' \circ e$ exists in $\mathcal{E}$. As usual, we define by $(\mathcal{E} \setminus S)^{op}$ the compositive graph *opposite* of $\mathcal{E} \setminus S$.

Theorem 3. *Let $\mathcal{E}$ be a projective sketch and let S be a realization of $\mathcal{E}$.*

i) $S \cong \mathrm{colim}_{E \in (\mathcal{E} \setminus S)^{op}} (\mathcal{Y}_{\mathcal{E}}(E))$,

ii) Let $M \colon \mathcal{E} \longrightarrow \overline{\mathcal{E}}$ be a morphism of projective sketches and F_M its corresponding freely generating functor, then:

$$\mathrm{F}_M(S) \cong \mathrm{colim}_{E \in (\mathcal{E} \setminus S)^{op}} (\mathcal{Y}_{\overline{\mathcal{E}}}(M(E))) \ .$$

The first part of this theorem defines a density property: each realization can be seen as a colimit of realizations which are in the image of the Yoneda morphism. The second part gives us a method for building the freely generating functor associated to a morphism of projective sketches $M \colon \mathcal{E} \longrightarrow \overline{\mathcal{E}}$. Given a realization S of $\mathcal{E}$, $\mathrm{F}_M(S)$ is obtained by the following process. First, it is necessary to obtain, for each point C in $\mathcal{E}$, the image by the Yoneda morphism $\mathcal{Y}_{\overline{\mathcal{E}}}$ of $M(C)$. Then, $\mathrm{F}_M(S)$ is calculated as a colimit of copies of these images using as index the points E in $(\mathcal{E} \setminus S)^{op}$.

2.4 Diagrammatic Specification of Equational Logic

In the context of algebraic specifications, as for instance in [12], an equational specification is made up of three components: a set of sorts, a set of operations (on the set of sorts) and finally a set of equations. The set of sorts and the set of operations form the signature of the specification. These three sets are closely related. Some strings of sorts are used to introduce the operations, and some terms (derived from the operations) are used to introduce the equations.

For instance, an equational specification for semigroups E_{SGRP} consists of a signature which has one sort g and one operation $prd \colon g\ g \to g$. It uses the string $g\ g$. Besides, it has one equation $prd(a, prd(b, c)) = prd(prd(a, b), c)$, where a, b, c are variables of sort g. The models of this specification are the semigroups.

The equation of the above specification can be written without variables, by using relations between composite arrows [1]: $prd \circ fact(p_1, prd) \equiv prd \circ fact(prd, p_2)$, with two projection arrows $p_i \colon g\ g \to g$, $i = 1, 2$ and one factorization arrow $fact$. Then, that equational specification can be replaced with a projective sketch with some parallel arrows (arrows which share sources and targets) as equations (see again [1]). The idea is that set-valued realizations of this sketch coincide with the models of the specification.

In the case of semigroups, we consider the sketch whose set of points is $\{g, g\ g\}$, and whose set of arrows is $\{prd, p_1, p_2, fact(p_1, prd), fact(prd, p_2), prd \circ$

$fact(p_1, prd), prd \circ fact(prd, p_2)\}$. The source and target arrows are defined from the operations in the natural way (source(prd)=g g, target(prd)=g). Besides, a distinguished cone is included in order to establish that g g will represent a binary product: a cone with vertex g g and whose base is the discrete diagram $\{g, g\}$. Moreover, a pair of parallel arrows have to be included in order to represent the equation $prd \circ fact(p_1, prd) \equiv prd \circ fact(prd, p_2)$.

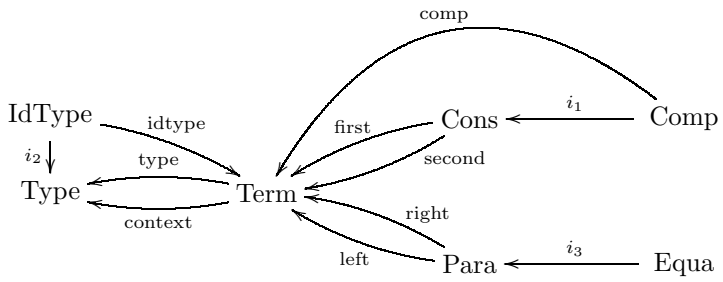
In general, an equational specification S can be represented as a compositive graph together with:
– some cones $(p_i\colon P \to X_i)_{i=1,\ldots,n}$, called *(potential) products*, whose vertex P is usually denoted by $X_1 \times \ldots \times X_n$. When $n = 0$, the vertex P of the cone with empty base is called a *(potential) terminal point* and it is denoted by $\mathbb{1}$,
– some pairs of parallel arrows $(f\colon X \to Y, g\colon X \to Y)$ in S, called *equations* (or *potential equalities*), which are written $f \equiv g$.

Note that this representation of an equational specification is slightly different from the usual one. In this new setting, a point may correspond to a sort or to a list of sorts, and an arrow to an operation or to a term.

Now, our aim is to use projective sketches at the meta-level in order to obtain a diagrammatic specification of equational logic. Note that, in this context, an equational specification will be a realization of that sketch. The starting point of this process is the sketch for compositive graphs $\mathcal{E}_{Comp}$. This sketch is going to be enriched and we are going to obtain a new sketch $\mathcal{E}_{Eq}$ which allows to cover equations and products. In our attempt, only constants, unary and binary operations are going to be represented, but it is clear that the process can be extended to general products. Then, a propagator $P_{Eq}\colon \mathcal{E}_{Eq} \longrightarrow \overline{\mathcal{E}}_{Eq}$ will be built, in such a way that the P_{Eq}-specifications are the *equational specifications*, the P_{Eq}-domains are called the *equational categories*, and the propagator P_{Eq} corresponds to the equational logic.

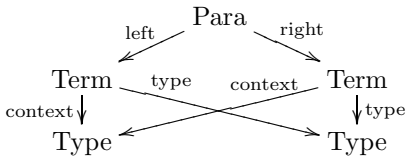
Equations

In order to include equations we build the following graph:

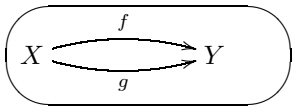


So, this projective sketch $\mathcal{E}_{Eq}$ is an enrichment of $\mathcal{E}_{Comp}$. The points Para and Equa stand for "pairs of parallel arrows" and "equations", respectively. The arrows left and right extract the two terms of an equation and $i_3\colon$ Equa $\rightarrowtail$ Para is defined as a monomorphism (an equation is a pair of parallel terms).

The projective sketch $\mathcal{E}_{Eq}$ also contains the distinguished cone with vertex Para which establishes that Para represents pairs of parallel terms:



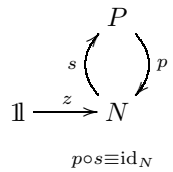
By using the Yoneda properties, we can give the following description of the point Para:



Now, it is possible to define a new sketch $\overline{\mathcal{E}}_{Eq}$ for *equational categories*. We can build $\overline{\mathcal{E}}_{Eq}$ by enriching $\mathcal{E}_{Cat}$ in a similar way to the process used to obtain $\mathcal{E}_{Eq}$ from $\mathcal{E}_{Comp}$ (indeed, an equational category is defined in [8] as a category with a congruence relation and in that paper some additional rules are needed to obtain that relation). Then, the *equational propagator* is the enrichment $P_{Eq}\colon \mathcal{E}_{Eq} \longrightarrow \overline{\mathcal{E}}_{Eq}$. Hence, P_{Eq}-specifications are equational specifications and P_{Eq}-domains are equational categories. In this paper, we are only interested in the category *Set*, i.e. the category of sets, which has the equality of maps as congruence relation. This category will be considered as P_{Eq}-domain for all P_{Eq}-specifications. But of course, it is possible to work with other types of equations in other contexts.

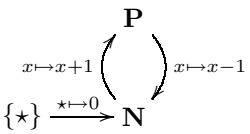
Example 9. Let us consider the diagrammatic specification T which has three points V, N and P, where V is a terminal point, five arrows $z\colon V \longrightarrow N$, $s\colon N \longrightarrow P$, $p\colon P \longrightarrow N$, $\mathrm{id}_N\colon N \longrightarrow N$ and $p \circ s\colon N \longrightarrow N$, where id_N is the identity of N and $p \circ s$ is the composite of s and p, and one equation $p \circ s \equiv \mathrm{id}_N$. In the illustration, the identity and the composite have been omitted, and we use the symbol $\mathbb{1}$ to represent that V is a terminal point:

T :

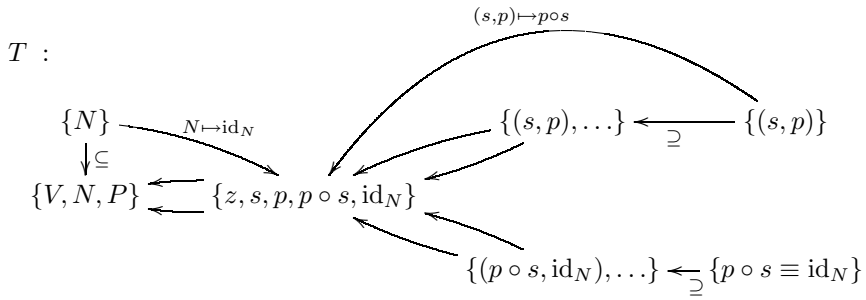


The specification T has a very natural model M which interprets the sort V as a singleton $\{\star\}$, the sort N as the set $\mathbf{N}$ of non-negative integers, the sort P as the set $\mathbf{P}$ of positive integers, the constant z as the constant map $zero\colon \star \mapsto 0$, and the operations s and p as the maps $suc\colon x \mapsto x + 1$ and $pre\colon x \mapsto x - 1$, respectively:

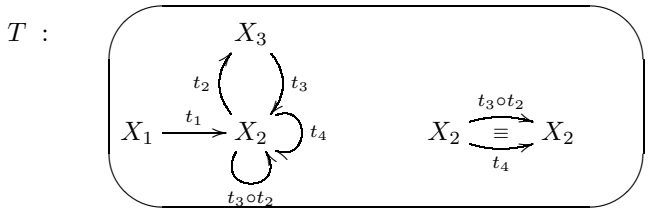
M :



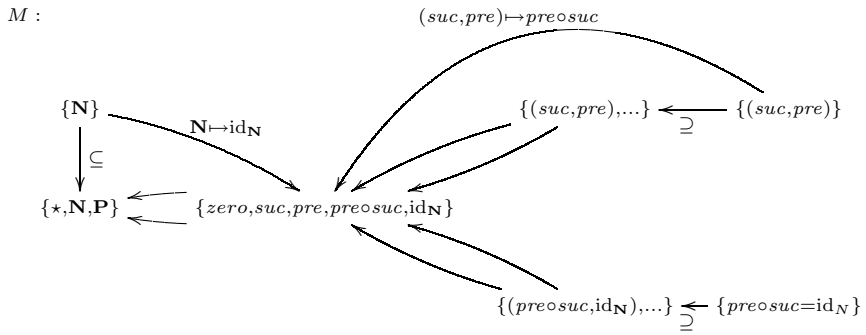
On the one hand, T is a realization of $\mathcal{E}_{Eq}$, i.e. a P_{Eq}-specification for the equational propagator $P_{Eq}\colon \mathcal{E}_{Eq} \longrightarrow \overline{\mathcal{E}}_{Eq}$, which gives a completely different illustration for T:



On the other hand, this realization T of $\mathcal{E}_{Eq}$ can be seen as the colimit of three copies of $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Eq})}(\mathrm{Type})$ (which has the same representation as $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Gr})}(\mathrm{Type})$ in Example 8), one copy for each element of $T(\mathrm{Type})$, and five copies of $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Eq})}(\mathrm{Term})$ (which has the same representation as $\mathcal{Y}_{\mathrm{Proto}(\mathcal{E}_{Gr})}(\mathrm{Type})$ in Example 8), one copy for each element of $T(\mathrm{Term})$, and so on. If they are summed following the diagram defined by $(\mathcal{E}_{Eq} \setminus T)^{op}$, we retrieve the representation:



Finally, *Set* is a realization for $\overline{\mathcal{E}}_{Eq}$, i.e. a P_{Eq}-domain for the equational propagator $P_{Eq}\colon \mathcal{E}_{Eq} \longrightarrow \overline{\mathcal{E}}_{Eq}$. Then, a P_{Eq} model of T with values in *Set* is a homomorphism in $\mathrm{Hom}_{\mathcal{S}pec(P_{Eq})}(T, \mathrm{G}_{P_{Eq}}(Set))$, i.e. a family of applications $f_E\colon T(E) \longrightarrow \mathrm{G}_{P_{Eq}}(Set)(E)$, one application for each point of E in $\mathcal{E}_{Eq}$. In these families, the application f_{Type} assigns each type of T to a set, and the application f_{Term} assigns each term of T to an application and so on. For instance, the model M has the following interpretation:

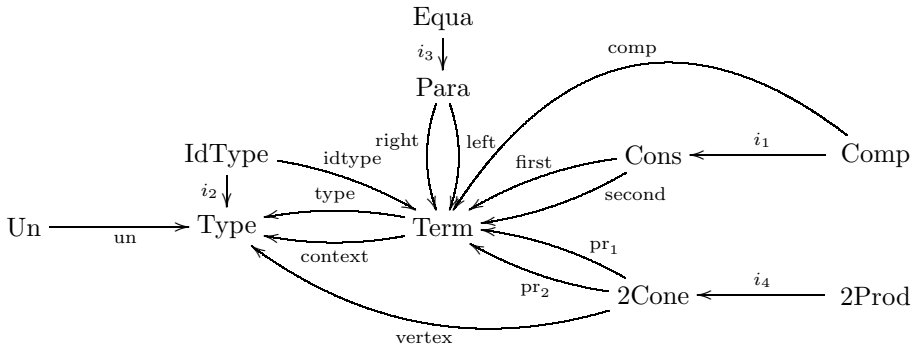


It must be observed that the terms of the specifications for $\mathcal{E}_{Eq}$ have exactly one argument. We can easily avoid this problem by considering n-ary terms as terms which have a n-ary record type $(X_1, \ldots, X_n)$ as unique argument (this record type is considered a product of types). But, at this point, it is not possible to distinguish between an operation with a "single" sort as argument from an operation which has a n-ary product of sorts as argument. In the following subsection products are included in our realizations.

Products

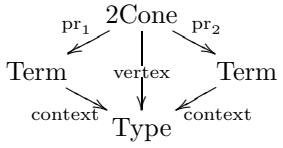
The sketch $\mathcal{E}_{Eq}$ is now enriched in order to include binary products of sorts and a sort that allows to represent the constants. This sketch will be also denoted by $\mathcal{E}_{Eq}$.

A binary product is a binary cone which satisfies a universal property. So, we include in the sketch two new points: 2Cone for binary cones and 2Prod for binary products. Similarly, a point Un is included in order to specify the constants:



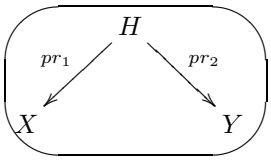
The arrows vertex, pr_1, pr_2 stand for the vertex and for the two projections of a binary cone. The arrow i_4 represents the inclusion of binary products as binary cones.

Moreover, the projective sketch $\mathcal{E}_{Eq}$ contains a distinguished cone with vertex Un and empty base (which determines that realizations for this sketch will have a singleton on this point), and a distinguished cone with vertex 2Cone:



which establishes that a binary cone consists of two terms with the same context.

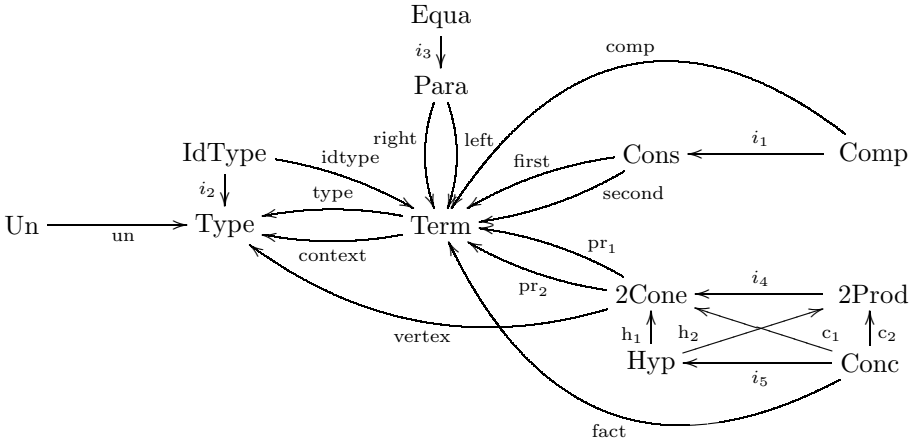
By using the Yoneda properties, we can give the following description for 2Cone:



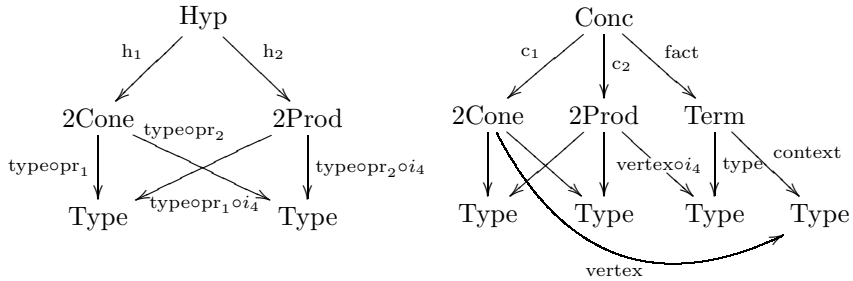
Now, it is necessary to impose the universal property of binary products: if P is a binary product then, for each binary cone C having the same base of P, there exists a (unique) factorization term from the vertex of C to the vertex of P whose composites with the projections of P commute with the projections of C.

The property of existence of this factorization term can be visualized as a rule with a hypothesis and a conclusion. The hypothesis consists of a binary product and a binary cone with the same base. The conclusion consists of a binary product, a binary cone (with the same base) and a term from the vertex of the cone to the vertex of the product which commutes with the projections.

In order to represent this property, two new points Hyp and Conc are included in the projective sketch $\mathcal{E}_{Eq}$:



The projective sketch $\mathcal{E}_{Eq}$ also contains a distinguished cone with vertex Hyp which determines that the hypothesis consists of a binary cone and a binary product such that they share their basis, and a distinguished cone with vertex Conc which determines that the conclusion consists of such binary cone and binary product and a term from the vertex of the cone to the vertex of the product:



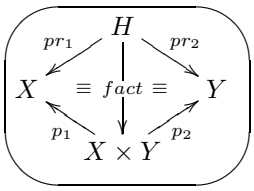
Now, we still have to impose that the composites of the factorization term with the projections of the product are the projections of the cone. In order to include these composites, a new arrow for the first projection $\mathrm{com}_1 \colon \mathrm{Conc} \longrightarrow \mathrm{Comp}$ is added to $\mathcal{E}_{Eq}$ (a similar one is needed for the second projection). This arrow is needed to define the composite of fact and $\mathrm{pr}_1 \circ i_4 \circ \mathrm{c}_2$ as a term. This is obtained through the following equalities:

$$\mathrm{fact} = \mathrm{first} \circ i_1 \circ \mathrm{com}_1 \quad \text{and} \quad \mathrm{pr}_1 \circ i_4 \circ \mathrm{c}_2 = \mathrm{second} \circ i_1 \circ \mathrm{com}_1 \ ,$$

and then the composition of these terms are equal to the first projection of the cone:

$$\mathrm{pr}_1 \circ \mathrm{c}_1 = \mathrm{comp} \circ \mathrm{com}_1 \ .$$

The description given by Yoneda properties of the conclusion Comp of this rule is:



Similar processes allow to represent the unicity of the factorization arrow and to model that the point Un represents a terminal point.

Now, rules are declared by enriching $\mathcal{E}_{Eq}$ with inverses. For instance, an arrow $i_5^{-1} \colon \mathrm{Conc} \longrightarrow \mathrm{Hyp}$ is added, such that:

$$i_5 \circ i_5^{-1} = \mathrm{id}_{\mathrm{Hyp}} \qquad \text{and} \qquad i_5^{-1} \circ i_5 = \mathrm{id}_{\mathrm{Conc}} \ .$$

3 The *imp* Construction

We have pointed out that in Kenzo and EAT systems [7, 14] two *layers* of data structures coexist. In a first layer, one finds the usual data structures like integer numbers, (finite) lists, trees of symbols (to represent linear combinations or polynomials) and so on. In the second layer, one must deal with algebraic structures like (graded) groups, rings, simplicial sets or chain complexes, whose elements are data belonging to the first layer. These algebraic structures are first order elements in Algebraic Topology and then systems such as Kenzo and EAT

do not work with a unique instance of these structures, but handle families of them at runtime.

To carry out the specification of this kind of structures, an operation between abstract data types was defined in [10]. This operation models the step from working with a data type to working with the data type of families of "elements" of that data type. The operation is called *imp* construction (this name has been chosen because this kind of specifications are related to implementations of structures rather to the structures themselves, i.e. related to the treatment at low level that the systems make of the structures).

The following simple example is used to explain the syntactic aspects of this construction. Let E_{SGRP} be the equational specification for semigroups. It consists of a signature Σ_{SGRP} which has one sort g and one operation $prd\colon g\ g \to g$. Besides, it has one equation $prd(a, prd(b, c)) = prd(prd(a, b), c)$, where a, b, c are variables of sort g.

The specification E_{SGRP} is obviously the basis of the algebraic specification for a semigroup, whose underlying set is abstracted by the sort g. But if, as it is usual in symbolic computation systems, it is necessary to handle several semigroups on the same underlying data set, a new data type, which remains hidden in the signature Σ_{SGRP}, must be considered: the type of semigroups represented on g. If we make explicit this invisible (or hidden) type, we obtain a new specification, denoted by $E_{SGRPimp}$, whose signature $\Sigma_{SGRPimp}$ contains a new sort $imp_{\Sigma_{SGRP}}$, and an operation: $imp\_prd\colon imp_{\Sigma_{SGRP}}\ g\ g \to g$. Besides, it has one equation: $imp\_prd(z_{imp}, a, imp\_prd(z_{imp}, b, c)) = imp\_prd(z_{imp}, imp\_prd(z_{imp}, a, b), c)$, where a, b, c are variables of sort g and z_{imp} is a variable of the new sort.

The $\Sigma_{SGRPimp}$-algebras for $E_{SGRPimp}$ represent families of Σ_{SGRP}-algebras for E_{SGRP}, in the sense that each element of the carrier set for the distinguished sort allows to retrieve a Σ_{SGRP}-algebra which satisfies the equation. To be more precise, given a $\Sigma_{SGRPimp}$-algebra A, each element $a \in A_{imp_{\Sigma_{SGRP}}}$ defines the Σ_{SGRP}-algebra $A_a = \langle A_g, imp\_prd(a, -, -)\rangle$. Note that the functions of A_a are obtained by fixing the element a as the first argument of the functions in A.

In general, given a specification $E = (\Sigma, \mathrm{E})$ with $\Sigma = (S, \Omega)$, a new specification $E_{imp} = (\Sigma_{imp}, \mathrm{E}_{imp})$ with $\Sigma_{imp} = (S_{imp}, \Omega_{imp})$ can be defined as follows:

- $S_{imp} = S \cup \{imp_\Sigma\}$ with $imp_\Sigma \notin S$,
- for each operation $\omega\colon s_1 \dots s_n \to s$ in Ω, an operation $imp\_\omega\colon imp_\Sigma s_1 \dots s_n \to s$ is included in Ω_{imp},
- for each equation $t = s$ in E, an equation $\theta(t) = \theta(u)$ is included in E_{imp} where $\theta = (\theta_s)_{s \in S}$ is a family of maps between the terms of Σ and the terms of Σ_{imp} defined as follows:
  - if $t = x$, a variable of sort $s \in S$, then $\theta_s(x) = x$;
  - if $t = \omega(t_1, \dots, t_n)$, with $\omega\colon s_1 \dots s_n \to s \in \Omega$, $n \geq 0$ and t_i term of Σ $i = 1, \dots, n$, then $\theta_s(\omega(t_1, \dots, t_n)) = imp\_\omega(z_{imp}, \theta_{s_1}(t_1), \dots, \theta_{s_n}(t_n))$, with z_{imp} a variable of sort imp_Σ.

In the following section we are going to describe this construction using diagrammatic techniques.

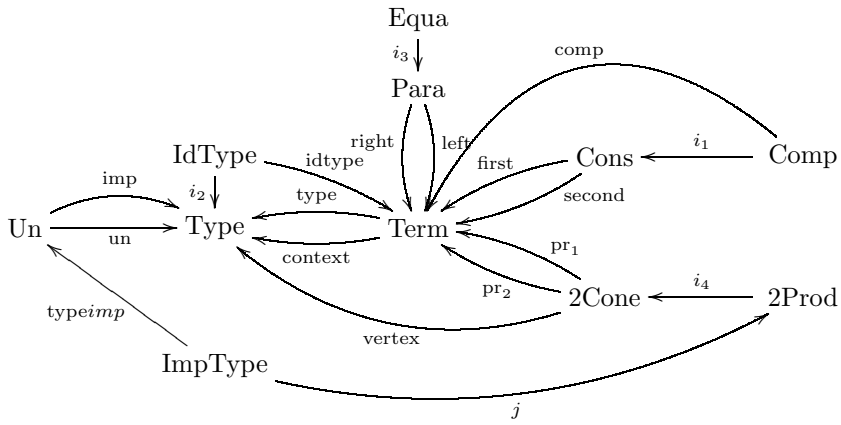
4 Diagrammatic Specification of the *imp* Construction

A diagrammatic specification of the *imp* construction will be given in this section in terms of a freely generating functor associated to a sketch morphism. The source of this sketch morphism will be the sketch $\mathcal{E}_{Eq}$ for equational specifications in Section 2.4, its target will be the sketch $\mathcal{E}^*_{Eq}$ for *pointed* equational specifications. Roughly speaking, a pointed specification consists of an equational specification such that a sort is distinguished in its signature.

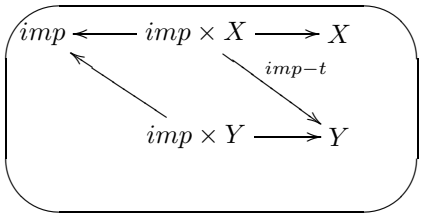
Our stating point is the compositive graph $\mathcal{E}_{Eq}$. Then, the graph $\mathcal{E}^*_{Eq}$ is defined as an extension of it. In order to obtain pointed specifications, a new arrow is added to the initial graph:

$$\text{imp}\colon \text{Un} \longrightarrow \text{Type}\ .$$

This arrow determines the distinguished sort. Now, it is possible to generate the binary products having this sort as first component. These products are represented through a new point ImpType (an inclusion $j\colon \text{ImpType} \rightarrowtail \text{2Prod}$ and a new arrow type*imp*$\colon \text{ImpType} \longrightarrow \text{Un}$ are also included in the graph):

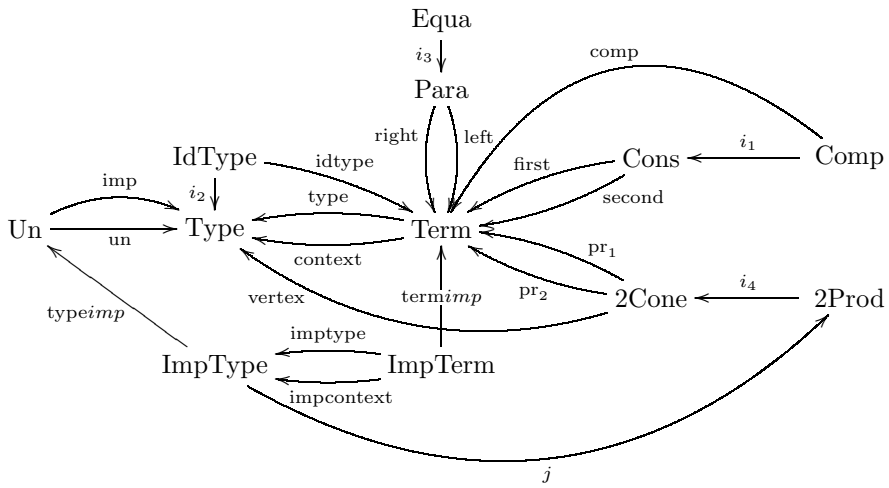


In a similar way, we have to state explicitly the point ImpTerm of terms whose context is a type in ImpType. Moreover, the corresponding arrows impcontext and imptype have to be included. For a given realization S of $\mathcal{E}^*_{Eq}$, an element of $S(\text{ImpType})$ will be a binary product: $(imp \longleftarrow imp \times X \longrightarrow X)$ and elements of $S(\text{ImpTerm})$ are arrows:



We say that $imp - t$ is the term associated to the *imp* term.

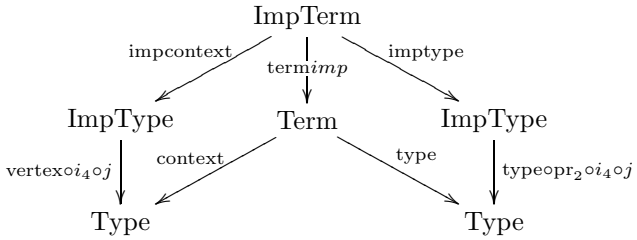
An arrow, termimp: ImpTerm $\longrightarrow$ Term, which extracts the associated term, is also included:



It is clear that some new composites and distinguished cones have to be considered. For instance, in order to establish that the first component of an ImpType is the distinguished type we have to include the composite:

$$\mathrm{type} \circ \mathrm{pr}_1 \circ i_4 \circ j = \mathrm{imp} \circ \mathrm{type}imp \ .$$

and the property of the term associated to an *imp* term which has as context a type in ImpType is obtained through the distinguished cone:



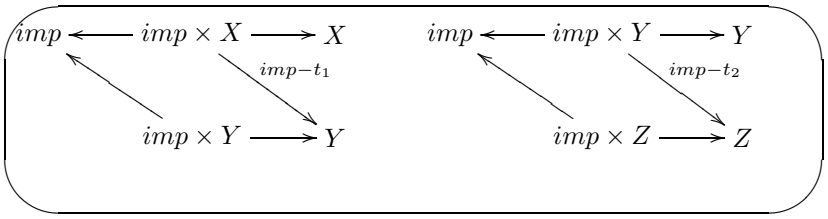
In the next subsections we are going to briefly introduce the rest of *imp* components which must be included in $\mathcal{E}^*_{Eq}$. To preserve the readability of the paper, some technical details of these construction will be avoided.

4.1 Composites and Identities

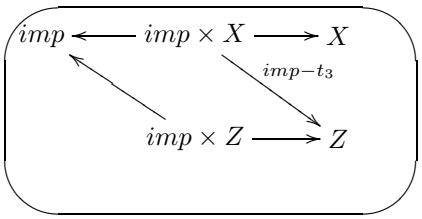
In order to represent the *imp* components associated to the composition of arrows we have to enrich the sketch $\mathcal{E}^*_{Eq}$ with two points ImpCons and ImpComp (for pairs of consecutive and composable *imp* terms, respectively). We have also to consider the corresponding projections (impfirst and impsecond from ImpCons

to ImpTerm), the inclusion from ImpComp to ImpCons and, finally, an arrow to extract the *imp* term associated to the composition of two composable *imp* terms (impcomp from ImpComp to ImpTerm). Then, composites and distinguished cones on these *imp* points and *imp* arrows are included in $\mathcal{E}^*_{Eq}$ which correspond to composites and distinguished cones in $\mathcal{E}_{Eq}$.

If we consider a realization of $\mathcal{E}^*_{Eq}$ (that is, a pointed signature) and two of its composable *imp* terms:



The composite must be an *imp* term



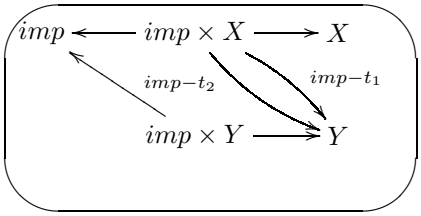
It is clear that it is not possible to directly use the composition of terms (in fact, they are not composable terms). The *imp* composite of these *imp* terms is defined by the composite of $fact(pr_1, imp - t_1)\colon imp \times X \longrightarrow imp \times Y$ and $imp - t_2\colon imp \times Y \longrightarrow Z$, where $fact$ is the factorization term obtained from the universal property of the binary product $imp \times Y$ (by using the cone $imp \times X$ with arrows pr_1, its first projection, and $imp - t_1$). This idea is (a variant of) a well-know construction in Category Theory, called *Kleisli composition* [11]. We are not going to develop the technical details which are needed in order to give a complete formalization of the above definition.

With regard to identities, a new point ImpIdType, and the corresponding arrows from it to ImpTerm and ImpType, have to be included in the graph.

4.2 Equations

Similarly to composite terms, the *imp* equations are included in the sketch $\mathcal{E}^*_{Eq}$ by means of two points: ImpPara and ImpEqua, which represent parallel *imp* terms and *imp* equations, respectively. The two corresponding projections from ImpPara to ImpTerm and the inclusion of ImpEqua into ImpPara are also considered. Moreover, we have to impose (by means of the corresponding distinguished cones and composites) that the description of a pair of parallel *imp* terms (for

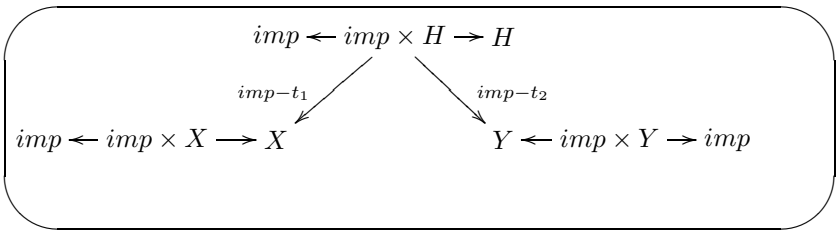
a given realization) is the following:



Then, two new arrows have to be pointed out in the graph in order to extract the parallel terms (the equation, respectively) associated to an *imp* parallel term (*imp* equation, respectively).

4.3 Cones and Products

Finally, the *imp* components which correspond to binary cones and binary products are represented. This leads us to introduce two new points Imp2Cone and Imp2Prod and the corresponding *imp* arrows and distinguished cones. For instance, the *imp* binary cone (for a realization of $\mathcal{E}^*_{Eq}$) is described as follows:



where the points *imp* should be identified.

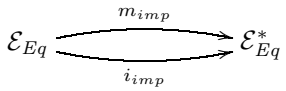
4.4 Two Sketch Morphisms to Represent the *imp* Construction

In the above subsections, a sketch $\mathcal{E}^*_{Eq}$ for pointed equational specifications have been built by enriching the sketch $\mathcal{E}_{Eq}$ for equational specifications. Then, a first morphism of sketches can be trivially defined between these sketches: the inclusion morphism $i_{imp}\colon \mathcal{E}_{Eq} \longrightarrow \mathcal{E}^*_{Eq}$.

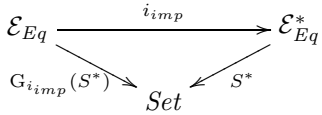
In order to obtain a diagrammatic representation of the *imp* construction, another morphism between these sketches is defined. Recall that the *imp* construction assigns to each operation in a signature a new operation which includes a new distinguished sort as first argument in its context. When a signature is represented as a projective sketch (following the comments in Section 2.4), each term in the signature with a type as context has to be mapped to a new term with the corresponding *imp* type as context. This is obtained with the freely generating functor associated to a morphism of sketches $m_{imp}\colon \mathcal{E}_{Eq} \longrightarrow \mathcal{E}^*_{Eq}$. This morphism assigns each point of $\mathcal{E}_{Eq}$ to its corresponding *imp* point of $\mathcal{E}^*_{Eq}$ and each arrow of $\mathcal{E}_{Eq}$ to its corresponding *imp* arrow of $\mathcal{E}^*_{Eq}$. With respect to

equations, it must be observed that in the two terms of an equation, the first projection over a product type (with the distinguished type as first component of the product) is always included in the corresponding *imp* equation. Besides, this first projection is also included in each *imp* composite included of these terms. This corresponds, in equational logic, to include the same variable of the distinguished sort in the *imp* terms.

The diagrammatic specification of the *imp* construction is obtained through the two morphism of sketches previously defined:

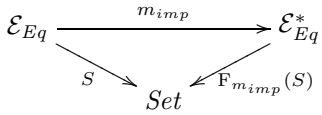


The inclusion morphism i_{imp} allows to obtain an equational specification from a pointed equational specification. To this aim, the omitting functor associated to this morphism is used:



where S^* is a set valued realization of $\mathcal{E}^*_{Eq}$ and $\mathrm{G}_{i_{imp}}(S^*) = S^* \circ i_{imp}$.

The morphism m_{imp} allows to build the *imp* specification associated to an equational specification. To this aim, the freely generating functor associated to this morphism is used:



where S is a set valued realization of $\mathcal{E}_{Eq}$.

Then, by construction, the following theorem is obtained:

Theorem 4. *Let S be a set valued realization of $\mathcal{E}_{Eq}$ which is the diagrammatic representation of an equational specification E, then $\mathrm{G}_{i_{imp}}(\mathrm{F}_{m_{imp}}(S))$ is the diagrammatic representation of E_{imp}.*

From the diagrammatic representation S of an equational specification, a diagrammatic representation of the corresponding *imp* specification is built by freely generation $\mathrm{F}_{m_{imp}}(S)$. This corresponds to a pointed equational specification. Then, this pointed equational specification can be seen as an equational specification by omitting the *imp* components.

5 Conclusions and Future Work

In this ongoing work we have translated into diagrammatic specifications our previous results in the specification of some data structures appearing in Sergeraert's symbolic computation systems which were obtained with standard algebraic specification techniques. In particular, we have given hints on the reason

why an important construction (called *imp* construction) in the specification of the systems, can be understood as a freely generating functor between suitable categories of diagrammatic realizations. Even if very partial, these positive results seem to indicate that this new kind of specification is promising in the field of symbolic computation.

This work may be continued along different lines. On the one hand, it will be necessary to continue the translation of our previous results into diagrammatic specifications: final algebra, hidden specifications, coalgebras, study of the inheritance... On the other hand, we may try to apply directly diagrammatic techniques to the specification of our systems and not only to make a translation of our previous work. In this line, the diagrammatic description of other parts of EAT and Kenzo systems should be undertaken. Finally, we can try to transfer our results to other symbolic computation systems.

References

1. M. Barr, Ch. Wells, Category Theory for Computer Science, Second Edition. Prentice Hall International, 1995.
2. L. Coppey, C. Lair, Leçons de Théorie des esquisses (I), Diagrammes, 12 (1984).
3. L. Coppey, C. Lair, Leçons de Théorie des esquisses (II), Diagrammes, 19 (1988).
4. C. Ehresmann, Introduction to the theory of structured categories. Report 10, University of Kansas, Lawrence (1966).
5. C. Domínguez, L. Lambán, V. Pascual, J. Rubio, Hidden specification of a functional system, in: R. Moreno-Díiaz, B. Buchberger, J.L. Freire (Eds.), Computer Aided Systems Theory (EUROCAST'2001), Lecture Notes in Computer Science, vol. 2178, Springer, Berlin, 2001, pp. 555–569.
6. C. Domínguez, J. Rubio, Modeling inheritance as coercion in a symbolic computation system, in: B. Mourrain (Ed.), International Symposium on Symbolic and Algebraic Computation (ISSAC'2001), ACM Press, 2001, pp. 107–115.
7. X. Dousson, F. Sergeraert, Y. Siret, The Kenzo program, Institut Fourier, Grenoble, 1999. Available at `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo`.
8. D. Duval, Diagrammatic Specifications, Mathematical Structures in Computer Science, 13(6) (2003) 857–890.
9. L. Lambán, V. Pascual, J. Rubio, An object-oriented interpretation of the EAT system, Applicable Algebra in Engineering, Communication and Computing, 14(3) (2003) 187–215.
10. L. Lambán, V. Pascual, J. Rubio, Specifying implementations, in: S. Dooley (Ed.), International Symposium on Symbolic and Algebraic Computation (ISSAC'99), ACM Press, 1999, pp. 245–251.
11. S. Mac Lane, Categories for the Working Mathematician, Springer-Verlag, 1971.
12. J. Loeckx, H.D. Ehrich, M. Wolf, Specification of Abstract Data Types, Wiley and Teubner, New York, 1996.
13. C. Lair, D. Duval, Esquisses et Spécifications, Manuel de Référence, 4ème partie: Fibrations et Eclatements, Lemmes de Yoneda et Modèles Engendrés, Rapport de Recherche du LACO 2001-03, 2003. Available at `http://www.unilim.fr/laco/rapports/`.
14. J. Rubio, F. Sergeraert, Y. Siret, EAT: Symbolic Software for Effective Homology Computation, Institut Fourier, Grenoble, 1997. Available at `ftp://fourier.ujf-grenoble.fr/pub/EAT`.