MAP2006, Mathematics, Algorithms and Proofs

Diagrammatic Specifications of Symbolic Computation Systems

C. Domínguez[1], D. Duval[2], L. Lambán[1] and J. Rubio[1] ([1]Universidad de La Rioja (Spain), [2]Université Joseph Fourier (France))

Abstract

The aim of this work is to describe the current state of an ongoing project to formalize, within 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. In a previous work, we gave some hints on the reason why a key construction (called *imp* construction) in the specification of the systems can be understood as a freely generating functor between suitable categories of diagrammatic realizations. Now, we will focus on the new open lines, giving a few details on the problems we have encountered up to now.

1 Introduction

Kenzo and its predecessor EAT 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 these systems, 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 work began the formal study of these programs in order to get a good understanding of their internal calculation processes [5, 4, 2]. 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 families of implementations of ADTs (several hundreds of *implementations* of the ADT group would populate the program memory). In [5], an operation called *imp* construction was defined. This construction models the step from a kind of structures to families of these structures.

2 The *imp* construction

We use the simple example of groups to explain this construction. Let Σ_{GRP} be the signature with one sort g and three operations:

$$\begin{array}{l} prd\colon g\ g \to g \\ inv\colon g \to g \\ unt\colon \ \to g \end{array}$$

The signature Σ_{GRP} is the basis of the algebraic specification for a group, 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 groups on the same underlying data set, a new data type, which remains hidden in the signature Σ_{GRP}, must be considered: the type of groups represented on g. If we make explicit this invisible (or hidden) type, we obtain a new signature, denoted by Σ_{GRPimp}, which contains a new sort $imp_{\Sigma_{GRP}}$ and the operations:

$$\begin{array}{l} imp\_prd\colon imp_{\Sigma_{GRP}}\ g\ g \to g \\ imp\_inv\colon imp_{\Sigma_{GRP}}\ g \to g \\ imp\_unt\colon imp_{\Sigma_{GRP}} \to g \end{array}$$

The Σ_{GRPimp}-algebras represent families of Σ_{GRP}-algebras in the sense that each element of the carrier set for the distinguished sort allows to retrieve a Σ_{GRP}-algebra.

3 Diagrammatic Logic

We need some definitions in order to briefly introduce diagrammatic logic [3].
Definition. A *compositive (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*. Besides, some points have an *identity arrow* and some *consecutive pairs* of arrows have a *composed arrow*.

Definition. A *projective sketch* is a compositive graph together with a set of *cones*. These cones are called *distinguished cones*.
A *morphism of projective sketches* is a morphism of compositive graphs which preserves the distinguished cones.

A projective sketch that represents the signature for groups is:

$$u \xrightarrow{unt} g \overset{inv}{\circlearrowright} \xleftarrow{prd} g \times g \qquad \text{with distinguished cones} \qquad u \to g \xleftarrow{pr_1} g \times g \xrightarrow{pr_2} g$$

Definition. A *(set-valued) realization* of a projective sketch maps each point to a set and each arrow to a map, 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.

Projective sketches can be used at the *meta-level* to define diagrammatic specifications.

Definition. With respect to a morphism of projective sketches $P\colon \mathcal{E} \longrightarrow \overline{\mathcal{E}}$:
- A *(diagrammatic) P-specification* is a realization of $\mathcal{E}$: $\mathcal{S}pec(P) = \mathcal{R}eal(\mathcal{E})$.
- A *(diagrammatic) P-domain* is a realization of $\overline{\mathcal{E}}$: $\mathcal{D}om(P) = \mathcal{R}eal(\overline{\mathcal{E}})$.
- It has associated an *omitting functor* $\mathrm{G}_P : \mathcal{D}om(P) \to \mathcal{S}pec(P)$.
- This omitting functor has a left adjoint $\mathrm{F}_P : \mathcal{S}pec(P) \to \mathcal{D}om(P)$ which is called *freely generating functor*.

Let S be a specification and $\mathcal{C}$ a domain for a morphism of sketches P. The *set of P-models of S with values in* $\mathcal{C}$, $\mathrm{Mod}_P(S, \mathcal{C})$, is defined as the set $\mathrm{Hom}_{\mathcal{D}om(P)}(\mathrm{F}_P(S), \mathcal{C})$ which is equivalent, by the adjunction property, to $\mathrm{Hom}_{\mathcal{S}pec(P)}(S, \mathrm{G}_P(\mathcal{C}))$.
For example, let us define a part of the projective sketch that can be used at the meta-level in order to obtain a diagrammatic specification of equational logic:

$$\mathcal{E}_{Eq}\ :\quad \mathrm{IdType} \xrightarrow{\mathrm{idtype}} \mathrm{Type} \underset{\mathrm{context}}{\overset{\mathrm{type}}{\rightleftarrows}} \mathrm{Term} \underset{\mathrm{pr}_2}{\overset{\mathrm{pr}_1}{\rightleftarrows}} \mathrm{2Cone} \xleftarrow{i} \mathrm{2Prod}, \quad \mathrm{vertex}: \mathrm{Term} \to \mathrm{2Prod}$$

Some distinguished cones are needed in order to guarantee that the points IdType and 2Cone will represent one-element sets and binary cones, respectively. Besides, some *rules* must be defined in order to ensure that the point 2Prod will represent binary products (see [3]).
A realization of this projective sketch that corresponds with the signature of groups assigns to IdType, Type, Term and 2Cone, the sets $\{1\}$, $\{g, g \times g, 1\}$, $\{prd, inv, unt, pr_1, pr_2\}$ and $\{g \leftarrow g \times g \rightarrow g\}$, respectively. The same set of 2Cone is assigned to 2Prod. Finally, obvious maps are assigned to arrows.

4 Diagrammatic specification of the *imp* construction

A diagrammatic specification of the *imp* construction was obtained in [1] in terms of a freely generating functor associated to a sketch morphism. In that work, a sketch for *pointed* equational specifications $\mathcal{E}^*_{Eq}$ was built. This implied adding to $\mathcal{E}_{Eq}$ a new arrow $\mathrm{imp}\colon \mathrm{Un} \longrightarrow \mathrm{Type}$ to *distinguish* a sort. Besides, for each point in $\mathcal{E}_{Eq}$, a new point was added to represent a new component in which the distinguished sort appeared. For instance, ImpType represented sorts like $imp \times X$ or ImpTerm terms like $imp - t\colon imp \times X \to Y$:

$$\mathcal{E}^*_{Eq}\ :\quad \mathrm{Un} \underset{\mathrm{un}}{\overset{\mathrm{imp}}{\rightleftarrows}} \mathrm{Type} \underset{\mathrm{context}}{\overset{\mathrm{type}}{\rightleftarrows}} \mathrm{Term} \underset{\mathrm{pr}_2}{\overset{\mathrm{pr}_1}{\rightleftarrows}} \mathrm{2Cone} \xleftarrow{i_4} \mathrm{2Prod}, \quad \mathrm{vertex}: \mathrm{Term} \to \mathrm{2Prod}$$
$$\mathrm{typeimp}: \mathrm{ImpType} \to \mathrm{Type}, \quad \mathrm{termimp}: \mathrm{ImpTerm} \to \mathrm{Term}$$
$$\mathrm{ImpType} \underset{\mathrm{impcontext}}{\overset{\mathrm{imptype}}{\rightleftarrows}} \mathrm{ImpTerm}, \quad j: \mathrm{ImpTerm} \to \mathrm{2Prod}$$

The diagrammatic specification of the *imp* construction was obtained through two morphism of sketches between $\mathcal{E}_{Eq}$ and $\mathcal{E}^*_{Eq}$. The first one, i_{imp}, was the inclusion morphism. The second one, m_{imp}, assigned each component of $\mathcal{E}_{Eq}$ to its corresponding *imp* component of $\mathcal{E}^*_{Eq}$.
Theorem. *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}.*

For instance, if GRP is the diagrammatic representation of the signature of groups, then a diagrammatic representation of the corresponding *imp* signature is built by free generation: $\mathrm{F}_{m_{imp}}(GRP)$. That is a pointed equational specification. Then, we obtain an equational specification by omitting the distinguished *imp* components.

5 Open lines

As we have indicated previously, the *imp* construction models the step from a kind of structures to families of these structures. Besides, working with implementations in [5], we obtained that the data structures which appear in EAT 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 [4], we reinterpreted our results in terms of object-oriented technologies such as hidden algebras or coalgebras. In this section, we will enumerate new open lines for the translation of our previous results into diagrammatic techniques with some problems or solutions we are encountering.

1. Parametrization and coalgebras. From the coalgebraic point of view, a family of data structures can be seen as a function $a\colon X \to A$ where A is the set of these data structures. This can also be interpreted as an indexed family where the set X acts as an index.

   Another interpretation arises from the parametrization process. This process is obtained from the diagrammatic specification of the composition of particular terms. Let X be a set, $f\colon X \to Y$ be a function and $a\colon U \to X$ be an element of the set X (U can be considered as the unit type, i.e. a singleton). Then, if the functions f and a are composed, we obtain an element of Y, $g = f \circ a\colon U \to X \to Y$. Therefore, if a signature Σ is considered as a specification $(U \to Y)$, then $(X \to Y)$ is precisely the pointed specification $\mathrm{F}_{m_{imp}}(\Sigma)$. Each element in X, $U \to X$, acts as a parameter in the family.

   For instance, for the group signature Σ_{GRP}, let us assume that exponentials are allowed, then the triple (prd, inv, unt) is an arrow $U \to Y$ where $Y = g^{g \times g} \times g^g \times g$. So, if $X = imp_{\Sigma_{GRP}}$, then the *imp*-signature for groups $\mathrm{F}_{m_{imp}}(\Sigma_{GRP})$ is obtained.
   Now, if we fix a data domain for all sorts except the distinguished sort, we recover the final object that we obtained in the coalgebraic framework for our very particular case. For instance, in the group case, if we fix the set D for g, and we consider the category of models for $\mathrm{F}_{m_{imp}}(\Sigma_{GRP})$ with $Y_D = D^{D \times D} \times D^D \times D$, then the final object has the set Y_D as set for the distinguished sort.

2. Hidden specification. We could try to define a sketch morphism in order to define a sketch for hidden specifications at a meta-specification level. In these specifications there are two different sets of sorts: hidden sorts and visible sorts. The first attempt consists in defining the following diagram:

   $$\mathcal{E}_{hid}\ :\quad \mathrm{Typev} \xrightarrow{i_v} \mathrm{Type} \underset{\mathrm{context}}{\overset{\mathrm{type}}{\rightleftarrows}} \mathrm{Term}, \quad \mathrm{Typeh} \xrightarrow{i_h} \mathrm{Type}$$

   which implies adding two new points for the two sets of sorts to the sketch of the equational specification. But, those sets should have a disjoint union. This condition can be represented in a sketch at the meta level, but that sketch would not be projective anymore.

   For our very particular case, we only have a distinguished sort (which could be considered as hidden) but much more study is necessary at this stage in order to include a more general case.

3. Implementation. A notion of implementation should be given using diagrammatic techniques in order to recover our first results. These results were based on Hoare's implementation notion. This point was not taken into account in our previous attempts using frameworks like coalgebras or hidden specifications and remains as future work.

References

[1] C. Domínguez, D. Duval, L. Lambán, J. Rubio, Towards diagrammatic specifications of symbolic computation systems. Seminar Mathematics, Algorithms and Proofs (MAP 2005), 2005.

[2] C. Domínguez, L. Lambán, J. Rubio, Object oriented institutions to specify symbolic computation systems. To appear in RAIRO - Theoretical Informatics and Applications.

[3] D. Duval, Diagrammatic Specifications. *Mathematical Structures in Computer Science*, 13(6) (2003) 857–890.

[4] 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.

[5] L. Lambán, V. Pascual, J. Rubio, Specifying implementations. In proceedings ISSAC'99, ACM Press, 1999, 245–251.