



Laboratoire d'Arithmétique, de Calcul formel et d'Optimisation  
ESA - CNRS 6090

---

# Mosaics for Specifications with Implicit State

Dominique Duval & Christian Lair

Rapport de recherche n° 2000-02

---

Université de Limoges, 123 avenue Albert Thomas, 87060 Limoges Cedex  
Tél. 05 55 45 73 23 - Fax. 05 55 45 73 22 - laco@unilim.fr

<http://www.unilim.fr/laco/>



# Mosaics for Specifications with Implicit State

Dominique Duval<sup>1</sup> and Christian Lair<sup>2</sup> — January 31, 2000

<sup>1</sup> Université de Limoges, LACO, 123 rue Albert Thomas, 87000 Limoges, France  
dominique.duval@unilim.fr, phone (33) 5 55 45 73 17, fax (33) 5 55 45 73 22

<sup>2</sup> Université Denis Diderot, U.F.R. de Mathématiques, 75251 Paris Cedex 05, France  
lairchrist@aol.com, phone (33) 1 44 27 54 10, fax (33) 1 44 27 54 10

**Abstract.** In this paper, we develop a new framework for algebraic specifications with implicit state. Among other results, this bridges the gap between specifications with implicit state and with explicit state.

## 1 Introduction

This paper is part of a project entitled *Sketches and Specification*.

An algebraic specification (or AS) [Astesiano *et al.* 99] is made of *sorts*, *operations* and *axioms*. It has *models* and *terms*, with a *congruence* relation on the terms. Each term can be *interpreted* in each model, and congruent terms have the same interpretation.

There are several ways to use AS's in order to deal with the notion of *state* in computer science. Roughly speaking, they can be divided in two classes, which we call, generically, *algebraic specifications with explicit state* (or AS-ES) and *algebraic specifications with implicit state* (or AS-IS). This terminology is borrowed from [Dauchy Gaudel 94], however here we use "AS-IS" in a more general sense.

On one hand, an AS-ES is simply an AS  $Spec^{expl}$  with a distinguished sort  $Q$ . Its meaning is given by a model of  $Spec^{expl}$ , where the sort  $Q$  is interpreted as the set  $\mathbb{Q}$  of states. An AS-ES does not require any generalization of the usual algebraic specification techniques. However, it is usually quite large and intricate, and its terms cannot be identified with the imperative programs.

On the other hand, an AS-IS does not include any sort for specifying the set of states. A review of various paradigms for AS-IS's, and related ones, can be found in [Gaudel *et al.* 99]. The terminology may differ significantly from one paper to another. In all paradigms, an AS-IS may be considered as an AS  $Spec^{impl}$  which is divided into several levels, according to their interaction with the state. The first levels define a subspecification of  $Spec^{impl}$ , which we call its *instant* subspecification, and a state is defined as a *model* of this instant subspecification. This is why the AS-IS's paradigms are usually called the *state-as-algebra* approach. The remaining levels define operations which act as functions upon the set of states.

Whereas its instant subspecification has relevant models, the complete AS  $Spec^{impl}$  is *meaningless*: it has no relevant model. However, its terms correspond to the imperative programs.

So, imperative programs can be identified with the terms of an AS  $Spec^{impl}$ , whereas their meaning is given by the models of *another* AS  $Spec^{expl}$ . From [guide2], this is characteristic of an *implicit* feature in a computer language. Then, using definitions and results from [guide2], we get a systematic method for building the AS-ES from the AS-IS, *i.e.* for building  $Spec^{expl}$  and its distinguished sort  $Q$  from  $Spec^{impl}$  and its decomposition in several levels. In addition, [guide2] also provides the required tools for *avoiding* this construction. Indeed, we may recover, directly from the AS-IS, the models of  $Spec^{expl}$ ; and we may get, directly from the AS-IS, a framework for the evaluation of imperative programs. Moreover, with this approach, we do not have to make any distinction between *elementary* and *non-elementary* operations.

In this paper, we provide a unified approach for dealing with AS-IS's. This approach is able to deal with many other implicit features in computer languages, like error handling (see [guide2]), overloading, coercions, subsorts, ... For this purpose, we introduce two new tools for specification: the *wefts* and the *mosaics*.

First, the *wefts* (for *trames*, in French), have been introduced by Lair in [Lair 87], and slightly generalized in [guide1]. They are based upon Ehresmann's *sketches* (for *esquisses*, in French) [Ehresmann 66, Ehresmann 68]. They are described in section 2. There is a category  $Weft(\mathcal{A})$  of  $\mathcal{A}$ -wefts for every category  $\mathcal{A}$ . Each usual AS may be identified to an *Ambi*-weft, where *Ambi* is the category of *ambigraphs* (which are defined in section 2.1, as oriented graphs with some additional structure).

Then, the *mosaics* have been designed (in [guide2]) to deal with implicit features of various kinds. They are described in section 3. Here, we only need some of them, which are called the *mosaics with implicit state*. Each AS-IS may be identified with a mosaic with implicit state. One of its components is an *Ambi*-weft  $\mathbf{S}$ , corresponding to  $Spec^{impl}$ . Another one is an  $\mathcal{A}_{St}$ -weft  $\mathbf{T}$ , for a category  $\mathcal{A}_{St}$  different from *Ambi*: this is the reason why we need  $\mathcal{A}$ -wefts for various categories  $\mathcal{A}$ . The decomposition of  $Spec^{impl}$  in several levels corresponds to a kind of homomorphism  $R$  (called a *stratification*) between  $\mathbf{T}$  and  $\mathbf{S}$ .

Some basic results about the mosaics with implicit state are presented in section 4. These results include the construction, from each mosaic with implicit state, of an *Ambi*-weft with the same meaning. So, this construction corresponds to the *explicitization* of the state.

In this paper, we do not assume any familiarity with the *Sketches and Specification* project. However, for comprehensive definitions and proofs, and for a deeper understanding of our methods, the interested reader might have a look at its *User's Guide* or *Reference Manual* papers [guide1, guide2, ref1, ref2]. For instance, the *constraints* play a major role in wefts and mosaics. They are defined in a *functorial* way, which is highly powerful. But this point of view is unusual in AS's, hence we have chosen to give only some hints about constraints in this paper.

## 2 Wefts

Wefts are introduced here, together with an example: an *Ambi*-weft for specifying the integers. Ambigraphs are defined in section 2.1. *Ambi*-wefts, which are similar to usual algebraic specifications, are considered in section 2.2. Other wefts will prove useful in section 3, this is why section 2.3 is devoted to general wefts. Projective sketches are special wefts, which will be used for meta-specification, they are defined in section 2.4.

### 2.1 Ambigraphs

An *ambigraph*  $\mathcal{G}$  is an oriented graph, made of *points*  $G$  and *arrows*  $g : G_1 \rightarrow G_2$ , together with *identity arrows*  $id_G : G \rightarrow G$  for some points  $G$ , *composed arrows*  $g_2 \circ g_1 : G_1 \rightarrow G_3$  for some pairs of consecutive arrows ( $g_1 : G_1 \rightarrow G_2, g_2 : G_2 \rightarrow G_3$ ), and *equations*  $g_l \equiv g_r : G_1 \rightarrow G_2$  for some pairs of arrows with the same domain and codomain ( $g_l : G_1 \rightarrow G_2, g_r : G_1 \rightarrow G_2$ ).

An ambigraph  $\mathcal{G}$  can be considered as a specification of a very simple kind. Indeed,  $\mathcal{G}$  has *terms*, made of the arrows obtained by adding to  $\mathcal{G}$  as many identity and composed arrows as possible; there is a *congruence* relation among these terms, which is generated by the equations. And  $\mathcal{G}$  has *set-valued realizations*, which associate to each point a set, to each arrow a map, to each identity or composed arrow an identity or composed map, and to each equation an equality of maps. In a given set-valued realization of  $\mathcal{G}$ , each term of  $\mathcal{G}$  can be interpreted in a natural way, and two congruent terms clearly have the same interpretation.

For instance, in order to specify the integers, we may use the following ambigraph.

AMBIGRAPH  $\mathcal{G}_{Int}$ :

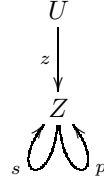
*points:*  $U, Z,$

*arrows:*  $z : U \rightarrow Z, s : Z \rightarrow Z, p : Z \rightarrow Z,$

*identity arrow:*  $id_Z : Z \rightarrow Z,$

*composed arrows:*  $p \circ s : Z \rightarrow Z, s \circ p : Z \rightarrow Z,$

*equations:*  $p \circ s \equiv id_Z : Z \rightarrow Z, s \circ p \equiv id_Z : Z \rightarrow Z.$



From now on, the identity and composed arrows, whenever they appear in the equations, are not explicitly mentioned.

Among the terms of  $\mathcal{G}_{Int}$  are  $s \circ z : U \rightarrow Z$  and  $p \circ s \circ s \circ z : U \rightarrow Z$ , which are congruent. The set-valued realization  $\omega_{Int}$  of  $\mathcal{G}_{Int}$  maps the point  $U$  onto the one-element set  $\mathbb{U} = \{*\}$ , the point  $Z$  onto the set of integers  $\mathbb{Z}$ , the arrow  $z$  onto the constant map  $0 : \mathbb{U} \rightarrow \mathbb{Z}$  such that  $* \mapsto 0$ , and the arrows  $s$  and  $p$  onto the *successor* and *predecessor* maps  $succ, pred : \mathbb{Z} \rightarrow \mathbb{Z}$ . The interpretation of both terms  $p \circ s \circ s \circ z$  and  $s \circ z$  by  $\omega_{Int}$  is the constant map  $1 : \mathbb{U} \rightarrow \mathbb{Z}$ .

However, ambigraphs are very poor specification tools. They are not even able to specify properly constants, nor  $n$ -ary maps for any  $n \geq 2$ . A much better specification tool is offered by the *Ambi*-wefts.

Note that ambigraphs are the points of a *category Ambi*, with the *homomorphisms of ambigraphs* (which are defined in a straightforward way) as arrows.

## 2.2 *Ambi-wefts*

A *weft of ambigraphs*, or *Ambi-weft*,  $\mathbf{S}$ , is made of a *support*  $Supp(\mathbf{S})$ , which is an ambigraph, and *constraints* (note that some other specification paradigms use the word “constraint” in a related, though different, meaning).

For instance, a *binary product constraint* can be denoted  $(G; p_1 : G \rightarrow G_1, p_2 : G \rightarrow G_2)$ . A set-valued realization  $\omega$  of  $Supp(\mathbf{S})$  satisfies this constraint if the set  $\omega(G)$  is the cartesian product  $\omega(G_1) \times \omega(G_2)$ , and the maps  $\omega(p_1) : \omega(G) \rightarrow \omega(G_1)$  and  $\omega(p_2) : \omega(G) \rightarrow \omega(G_2)$  are the canonical projections. Because of the fundamental property of the cartesian product, for each pair of arrows  $(s_1 : H \rightarrow G_1, s_2 : H \rightarrow G_2)$ , we are allowed to add one arrow  $fact(s_1, s_2) : H \rightarrow G$ , together with two equations  $p_1 \circ fact(s_1, s_2) \equiv s_1$  and  $p_2 \circ fact(s_1, s_2) \equiv s_2$ .

A *terminal point constraint* can be denoted  $U = 1$ , where  $U$  is a point of  $Supp(\mathbf{S})$ . A set-valued realization  $\omega$  of  $Supp(\mathbf{S})$  satisfies this constraint if the set  $\omega(U)$  is a *terminal set*, i.e. if there is exactly one map from any set towards  $\omega(U)$ ; it means that  $\omega(U)$  is a one-element set. Consequently, for each point  $G$  we may add one arrow  $fact_G : G \rightarrow U$ : this arrow must be interpreted as the unique map from  $\omega(G)$  towards  $\omega(U)$ . In addition, since  $\omega(U)$  is a one-element set, arrows from  $U$  to any point  $G$  stand for *constants of G*.

The *set-valued realizations* of  $\mathbf{S}$  are the set-valued realizations of  $Supp(\mathbf{S})$  which *satisfy* the constraints. The set of set-valued realizations of  $\mathbf{S}$  is denoted  $Real(\mathbf{S}, Set)$ .

An equational algebraic specification  $Spec$  can easily be identified with an *Ambi-weft*  $\mathbf{S}$  with product constraints: there is a point  $G_s$  of  $\mathbf{S}$  for each sort  $s$  of  $Spec$ , as well as a point  $G_{s_1 \dots s_n}$  for each sequence  $s_1 \dots s_n$  of sorts which occurs as the domain of an operation of  $Spec$ . If  $n=0$ , there is a terminal point constraint  $G_\lambda = 1$  (where  $\lambda$  denotes the empty sequence of sorts), and if  $n \geq 2$ , there is an  $n$ -ary product constraint with vertex  $G_{s_1 \dots s_n}$  and projection arrows from  $G_{s_1 \dots s_n}$  to  $G_{s_i}$  for each  $i$  from 1 to  $n$ . Then, each operation  $f : s_1 \dots s_n \rightarrow s$  of  $Spec$  gives rise to an arrow  $g_f : G_{s_1 \dots s_n} \rightarrow G_s$  of  $\mathbf{S}$ , and each equation  $f_l(x_1, \dots, x_n) = f_r(x_1, \dots, x_n)$  of  $Spec$  to an equation  $g_{f_l} \equiv g_{f_r}$  of  $\mathbf{S}$ . The models of  $Spec$  can then be identified with the set-valued realizations of  $\mathbf{S}$ .

This correspondence can be generalized to non-equational algebraic specifications, in a less straightforward way. This will barely be used in this paper. A non-equational algebraic specification  $Spec$  still corresponds to an *Ambi-weft*  $\mathbf{S}$ : the support of  $\mathbf{S}$  remains an ambigraph, and the non-equational part of  $Spec$  is essentially expressed by more sophisticated constraints.

So, the constraints play a major role in the definition of the wefts. In an *Ambi-weft*, a constraint is defined from ambigraphs and homomorphisms of ambigraphs. The precise definition of a constraint, which can be found in [guide2], is *functorial*. This means that, roughly speaking: *as soon as something is well defined on ambigraphs, it is automatically well defined on Ambi-wefts*. Hence, a careful treatment of a poor specification tool (the ambigraphs) easily leads to a treatment of a much more powerful specification tool (the *Ambi-wefts*).

Like an algebraic specification, an *Ambi-weft*  $\mathbf{S}$  has *terms*, which are built from the terms of the ambigraph  $Supp(\mathbf{S})$  and from additional arrows given by

the constraints. The *congruence* relation among these terms extends the congruence among the terms of the ambigraph  $Supp(\mathbf{S})$ , taking into account the equations which may arise from the constraints. In a given set-valued realization of  $\mathbf{S}$ , each term of  $\mathbf{S}$  has an interpretation, and two congruent terms have the same interpretation.

For instance, we get a better specification of the integers with the *Ambi*-weft  $\mathbf{S}_{Int}$ , with support  $\mathcal{G}_{Int}$  and constraint  $U = 1$ . Among its terms are  $p \circ s \circ s \circ z : U \rightarrow Z$ ,  $s \circ z : U \rightarrow Z$ ,  $s \circ z \circ fact_Z \circ z : U \rightarrow Z$ , which are congruent. The set-valued realization  $\omega_{Int}$  of  $\mathcal{G}_{Int}$  satisfies the constraint, hence it is a set-valued realization of  $\mathbf{S}_{Int}$ . In the sequel, we will build various extensions of  $\mathbf{S}_{Int}$ , their set-valued realizations will always be assumed to extend  $\omega_{Int}$ .

**Fact .** *The algebraic specifications (and their models) can be identified with the Ambi-wefts (and their set-valued realizations).*

### 2.3 $\mathcal{A}$ -wefts

Actually, wefts can be built on top of various things, not only on top of ambigraphs. More generally, for any category  $\mathcal{A}$ , we have the following definition.

An  $\mathcal{A}$ -weft is made of a *support*, which is a point of  $\mathcal{A}$ , and *constraints*, which are defined from special configurations of points and arrows of  $\mathcal{A}$  (as explained in [guide1]).

In section 2.4 we consider *Comp*-wefts for a category *Comp* which is slightly different from *Ambi*. In section 3, when dealing with mosaics, we will need  $\mathcal{A}_{St}$ -wefts for a category  $\mathcal{A}_{St}$  which is fairly different from *Ambi*.

In addition, the *homomorphisms of  $\mathcal{A}$ -wefts* are defined as the arrows (of  $\mathcal{A}$ ) between the supports, which preserve the constraints: the image of a constraint must be a constraint of the same shape, for instance the image of a binary product constraint must be a binary product constraint. In this way, we get the category  $Weft(\mathcal{A})$  of  $\mathcal{A}$ -wefts. One major property of wefts is that this notion is *functorial*: if  $\Phi : \mathcal{A} \rightarrow \mathcal{A}'$  is a *functor* between two categories, then it can be canonically extended to a functor  $Weft(\Phi) : Weft(\mathcal{A}) \rightarrow Weft(\mathcal{A}')$  between the corresponding categories of wefts. Hence, roughly speaking: *as soon as something is well defined on a category  $\mathcal{A}$ , it is automatically well defined on the category  $Weft(\mathcal{A})$ .*

### 2.4 Projective sketches

Ehresmann's sketches [Ehresmann 66, Ehresmann 68] can be considered as wefts. A *compositive graph* is an oriented graph with identity and composed arrows (but no equation). The *limit constraints* generalize the binary product constraints and the terminal point constraints, like limits generalize binary products and terminal points in category theory. The category *Comp* of compositive graphs is defined in a straightforward way.

A *projective sketch*  $\mathbf{E}$  is a *Comp*-weft with only limit constraints, which are called the *distinguished projective cones* of  $\mathbf{E}$ .

We will use the projective sketches as “meta-specifications”, in the following meaning.

The set-valued realizations of a projective sketch  $\mathbf{E}$  are called its *models*. It is possible to define *homomorphisms of models of  $\mathbf{E}$* , in order to get a category  $\text{Mod}(\mathbf{E})$ . A category  $\mathcal{A}$  is *projectively sketchable* if it is equivalent to such a  $\text{Mod}(\mathbf{E})$ . Actually, many interesting categories are projectively sketchable; for instance, the category  $\text{Ambi}$  of ambigraphs is equivalent to  $\text{Mod}(\mathbf{E}_{\text{Ambi}})$  for a projective sketch  $\mathbf{E}_{\text{Ambi}}$ .

The projective sketch  $\mathbf{E}_{\text{Ambi}}$  includes points  $\text{Pt}$ ,  $\text{Ar}$ ,  $\text{Eq}$  and arrows  $\text{dom}$ ,  $\text{codom}$  :  $\text{Ar} \rightarrow \text{Pt}$ ,  $\text{proj}_l$ ,  $\text{proj}_r$  :  $\text{Eq} \rightarrow \text{Ar}$ . An ambigraph  $\mathcal{G}$  can be identified with a model (also denoted  $\mathcal{G}$ ) of  $\mathbf{E}_{\text{Ambi}}$  such that  $\mathcal{G}(\text{Pt})$  (resp.  $\mathcal{G}(\text{Ar})$ ,  $\mathcal{G}(\text{Eq})$ ) is the set of points (resp. of arrows, of equations) of  $\mathcal{G}$ , the map  $\mathcal{G}(\text{dom})$  (resp.  $\mathcal{G}(\text{codom})$ ) associates to each arrow  $g : G_1 \rightarrow G_2$  its domain  $G_1$  (resp. its codomain  $G_2$ ), and the map  $\mathcal{G}(\text{proj}_l)$  (resp.  $\mathcal{G}(\text{proj}_r)$ ) associates to each equation  $g_l \equiv g_r$  its left-hand side  $g_l$  (resp. its right-hand side  $g_r$ ). Let  $E$  be a point of  $\mathbf{E}_{\text{Ambi}}$ , and let  $g \in \mathcal{G}(E)$ , then  $g$  is called an *ingredient of  $\mathcal{G}$  of nature  $E$* : so, the points (resp. arrows, equations) of  $\mathcal{G}$  are the ingredients of  $\mathcal{G}$  of nature  $\text{Pt}$  (resp.  $\text{Ar}$ ,  $\text{Eq}$ ).

For instance, the ambigraph  $\mathcal{G}_{\text{Int}}$ , as a model of  $\mathbf{E}_{\text{Ambi}}$ , is such that  $\mathcal{G}_{\text{Int}}(\text{Pt}) = \{U, Z\}$ ,  $\mathcal{G}_{\text{Int}}(\text{Ar}) = \{z, s, p, id_Z, p \circ s, s \circ p\}$ ,  $\mathcal{G}_{\text{Int}}(\text{Eq}) = \{p \circ s \equiv id_Z, s \circ p \equiv id_Z\}$ . In addition,  $\mathcal{G}_{\text{Int}}(\text{dom})$  and  $\mathcal{G}_{\text{Int}}(\text{codom})$  map each arrow of  $\mathcal{G}_{\text{Int}}$  towards the point  $Z$ , except for  $\mathcal{G}_{\text{Int}}(\text{dom})(z) = U$ . The maps  $\mathcal{G}_{\text{Int}}(\text{proj}_l)$  and  $\mathcal{G}_{\text{Int}}(\text{proj}_r)$  assign to each equation of  $\mathcal{G}_{\text{Int}}$  its left-hand side and its right-hand side, respectively.

To recap,  $\mathcal{A}$ -wefts are defined for any category  $\mathcal{A}$ , and in practice  $\mathcal{A}$  is projectively sketchable, i.e.  $\mathcal{A} \simeq \text{Mod}(\mathbf{E})$  for a projective sketch  $\mathbf{E}$ . The *Ambi*-wefts, where  $\text{Ambi} \simeq \text{Mod}(\mathbf{E}_{\text{Ambi}})$ , are similar to algebraic specifications, with a more functorial treatment of constraints and axioms.

### 3 Mosaics with implicit state

The aim of this section is to introduce the mosaics for dealing with an implicit state. Mosaics in general are defined in section 3.1. A mosaic is made of three parts; two of them are common to all mosaics with implicit state, they are described in sections 3.3 and 3.5. The third part may vary, it is made of an *Ambi*-weft  $\mathbf{S}$ , an  $\mathcal{A}_{St}$ -weft  $\mathbf{T}$ , for a category  $\mathcal{A}_{St}$  different from  $\text{Ambi}$ , and a kind of homomorphism  $R$  between  $\mathbf{T}$  and  $\mathbf{S}$ . An example is described in sections 3.2 and 3.4.

#### 3.1 Mosaics

In [guide2], we have defined mosaics as specification tools for dealing in a proper way with implicit features of computer languages. In addition, we have defined the *ribbon* construction, for the *explicitization* of the implicit features: it builds a weft (called explicit), with the same meaning, from any given mosaic.

A mosaic is made of various wefts, which may be considered as the general patterns and the forms of the pieces, for the construction of the explicit weft.

The general pattern of the mosaic is given by two wefts: a weft  $\mathbf{S}$  corresponds to an approximate pattern, while another weft  $\mathbf{T}$ , which is obtained by annotating  $\mathbf{S}$ , corresponds to a detailed pattern. The annotations of  $\mathbf{T}$  give the precise references of the pieces which have to be used: all the pieces with a given reference  $r$  have the same form, which is described by a weft  $\kappa(r)$ .

More formally, from [guide2], given a category  $\mathcal{A}$ , an  $\mathcal{A}$ -mosaic  $\Sigma$  is made of:

- a homomorphism of projective sketches  $\rho : \mathbf{F} \rightarrow \mathbf{E}$  (the family of references),
- a counter-model  $\kappa$  of  $\mathbf{F}$  with values in  $Weft(\mathcal{A})$  (the family of forms of the pieces),
- a stratification  $R$  along  $\rho$ , i.e. a  $Mod(\mathbf{E})$ -weft  $\mathbf{S}$ , a  $Mod(\mathbf{F})$ -weft  $\mathbf{T}$ , and a loose homomorphism  $R : \mathbf{T} \rightarrow \mathbf{S} \circ \rho$  (the patterns).

The notions of counter-model and loose homomorphism will be explained below.

Dealing with a state is only one of the implicit features that may be handled by a mosaic. It corresponds to special mosaics, where the references and forms of pieces are fixed, as follows.

A *mosaic with implicit state* is a mosaic  $\Sigma$  such that:  $\mathbf{E} = \mathbf{E}_{Ambi}$  and  $\rho = \rho_{St} : \mathbf{F}_{St} \rightarrow \mathbf{E}_{Ambi}$  as in section 3.3,  $\mathcal{A} = \mathcal{A}_{Ambi}$  and  $\kappa = \kappa_{St}$  as in section 3.5.

Only the patterns may vary: a mosaic with implicit state is characterized by  $\mathbf{S}$ ,  $\mathbf{T}$  and  $R$ . An example of patterns is presented in sections 3.2 and 3.4.

### 3.2 The approximate pattern: an example

Here, we describe the approximate pattern  $\mathbf{S}_{Var}$  for a simple basic example: dealing with a variable  $x$  of type Integer, in the computer science meaning. In this example, the set of states  $\mathbb{Q}$  is endowed with a map  $value : \mathbb{Q} \rightarrow \mathbb{Z}$  which returns *the value of  $x$*  in a given state, and a map  $update : \mathbb{Z} \times \mathbb{Q} \rightarrow \mathbb{Q}$  for *updating the value of  $x$* . As usual, “updating” means that  $value(update(n, q)) = n$  for each integer  $n$  and each state  $q$ .

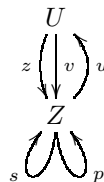
The approximate pattern  $\mathbf{S}_{Var}$  is a kind of specification for this situation, where the set of states is brutally “forgotten”.

*Ambi*-weft  $\mathbf{S}_{Var}$ :

*extends*:  $\mathbf{S}_{Int}$ ,

*arrows*:  $v : U \rightarrow Z$ ,  $u : Z \rightarrow U$ ,

*equation*:  $v \circ u \equiv id_Z$ .



The main interest of  $\mathbf{S}_{Var}$  lies in its terms, which correspond nicely to imperative programs. Indeed: *the terms of  $\mathbf{S}_{Var}$  mimic the imperative programs which we usually run when we deal with a variable  $x$* . For instance, the imperative instruction  $x := succ(x)$  corresponds to the term  $t = u \circ s \circ v : U \rightarrow U$  of  $\mathbf{S}_{Var}$ .

But the realizations of  $\mathbf{S}_{Var}$  are totally irrelevant. For instance, let us look at an instruction (like  $x := succ(x)$ ) which corresponds to a term  $t : U \rightarrow U$  of  $\mathbf{S}_{Var}$ . Then  $t \equiv id_U$ : indeed, because of the constraint  $U = 1$ , the interpretation

of  $t$  in any set-valued realization  $\omega_{Var}$  of  $\mathbf{S}_{Var}$  is the identity map of a one-element set. So, the map  $\omega_{Var}(t)$  does not do anything, whereas, of course, the instruction usually does something.

However, let us define the *instant subweft*  $\mathbf{S}_{Var}^{inst}$  of  $\mathbf{S}_{Var}$  as the subweft made of  $\mathbf{S}_{Int}$  and  $value : U \rightarrow Z$ . The set-valued realizations of  $\mathbf{S}_{Var}^{inst}$  are relevant. Indeed, let us assume that a state is characterized by the value of the variable  $x$ , i.e. that the map  $value : \mathbb{Q} \rightarrow \mathbb{Z}$  is a bijection. Then, the states can be identified with the set-valued realizations (extending  $\omega_{Int}$ ) of  $\mathbf{S}_{Var}^{inst}$ . Hence, the set of states can be recovered from  $\mathbf{S}_{Var}$ , even though there is no point in  $\mathbf{S}_{Var}$  for its specification. This is the starting point of the usual *state-as-algebra* paradigm.

### 3.3 The family of references

We have just seen that the realizations of  $\mathbf{S}_{Var}$  are irrelevant. Actually, for dealing with a variable of type Integer, we need some “interpretation of  $\mathbf{S}_{Var}$ ”, which will be defined in section 3.4. and which is not a realization of  $\mathbf{S}_{Var}$ .

More generally, in a mosaic  $\Sigma$ , we are not interested in the realizations of  $\mathbf{S}$ , but in some “interpretations of  $\mathbf{S}$ ”. The ingredients of  $\mathbf{S}$  may be classified, according to the required properties of their interpretations. When dealing with an implicit state, these properties are (essentially) among the properties listed below.

The interpretation of each point of  $\mathbf{S}$  must take into account the set of states. More precisely, it must be a set of the form  $X \times \mathbb{Q}$ , where  $\mathbb{Q}$  is the set of states.

Maps from  $X \times \mathbb{Q}$  to  $Y \times \mathbb{Q}$  are classified, as *static*, *access*, *modification* or *initialization* maps, according to the following definitions:

- a *static map* neither use nor modify the current state,
- an *access map* may give some information about the current state, but it is not allowed to modify it,
- a *modification map* may give some information about the current state, and it may also modify it,
- an *initialization map* builds a new state without using the current state.

Arrows of  $\mathbf{S}$  are classified in the same way, according to the properties which are required for their interpretations. It is worth noting that an access map *may* give some information about the current state, it does not *have* to. Hence any static arrow is an access arrow; in a similar way, any access arrow is a modification arrow and any initialization arrow is a modification arrow.

We also classify equations, as *static*, *access*, *strong* or *observation* equations, according to the following definitions:

- a *static* (resp. *access*, *strong*) equation  $s_l \equiv s_r$  is such that the arrows  $s_l$  and  $s_r$  are static (resp. access, any) arrows, and their interpretations are equal.
- an *observation* equation  $s_l \equiv s_r$  is such that the interpretations of  $s_l$  and of  $s_r$  are *observationally equal*: they return the same value, but may act differently upon the state.

Of course, any static equation is an access equation, any access equation is a strong equation, and any strong equation is an observation equation.

In order to deal with these properties, we introduce the following sets of *indices*:  $I_{\text{Pt}} = \{St\}$ ,  $I_{\text{Ar}} = \{st, acc, mod, ini\}$ ,  $I_{\text{Eq}} = \{st_{\equiv}, acc_{\equiv}, stg_{\equiv}, obs_{\equiv}\}$ . Each of these sets may be ordered, by  $i \Rightarrow j$  if property  $i$  implies property  $j$ . Then, for arrows:  $st \Rightarrow acc \Rightarrow mod$  and  $ini \Rightarrow mod$ , and for equations:  $st_{\equiv} \Rightarrow acc_{\equiv} \Rightarrow stg_{\equiv} \Rightarrow obs_{\equiv}$ .

From these indices, we may build a projective sketch  $\mathbf{F}_{St}$  “above”  $\mathbf{E}_{Ambi}$ , i.e. together with a homomorphism of projective sketches  $\rho_{St} : \mathbf{F}_{St} \rightarrow \mathbf{E}_{Ambi}$ . The points of  $\mathbf{F}_{St}$  above a point  $E$  of  $\mathbf{E}_{Ambi}$  are in bijection with the set of indices  $I_E$ . Hence, there are:

- one point  $[\text{Pt}, St]$  of  $\mathbf{F}_{St}$  above the point  $\text{Pt}$  of  $\mathbf{E}_{Ambi}$ ,
- four points  $[\text{Ar}, i]$  (for  $i \in I_{\text{Ar}}$ ) of  $\mathbf{F}_{St}$  above the point  $\text{Ar}$  of  $\mathbf{E}_{Ambi}$ ,
- four points  $[\text{Eq}, j]$  (for  $j \in I_{\text{Eq}}$ ) of  $\mathbf{F}_{St}$  above the point  $\text{Eq}$  of  $\mathbf{E}_{Ambi}$ .

The arrows, equations and distinguished projective cones of  $\mathbf{F}_{St}$  are defined in a similar way. For instance, there are four arrows  $[\text{dom}, i] : [\text{Ar}, i] \rightarrow [\text{Pt}, St]$  of  $\mathbf{F}_{St}$  above the arrow  $\text{dom} : \text{Ar} \rightarrow \text{Pt}$  of  $\mathbf{E}_{Ambi}$ , for  $i \in I_{\text{Ar}}$ , and similarly for  $\text{codom}$ . And there are four arrows  $[\text{proj}_l, j] : [\text{Eq}, j] \rightarrow [\text{Ar}, i(j)]$  of  $\mathbf{F}_{St}$  above the arrow  $\text{proj}_l : \text{Eq} \rightarrow \text{Ar}$  of  $\mathbf{E}_{Ambi}$ , for  $j \in I_{\text{Eq}}$  and  $i : I_{\text{Eq}} \rightarrow I_{\text{Ar}}$  such that  $i(st_{\equiv}) = st$ ,  $i(acc_{\equiv}) = acc$ ,  $i(stg_{\equiv}) = mod$  and  $i(obs_{\equiv}) = mod$ , and similarly for  $\text{proj}_r$ . It should be noted that the relations  $\Rightarrow$  on  $I_{\text{Ar}}$  and  $I_{\text{Eq}}$  correspond to arrows of  $\mathbf{F}_{St}$  which are not identity arrows, but which are above identity arrows of  $\mathbf{E}_{Ambi}$ : for instance, the relation  $st \Rightarrow acc$  corresponds to an arrow  $[id_{\text{Ar}}, st \Rightarrow acc] : [\text{Ar}, st] \rightarrow [\text{Ar}, acc]$  above the identity arrow  $id_{\text{Ar}} : \text{Ar} \rightarrow \text{Ar}$  of  $\mathbf{E}_{Ambi}$ .

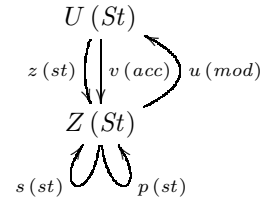
### 3.4 The detailed pattern: an example

Let  $\mathcal{A}_{St} = \text{Mod}(\mathbf{F}_{St})$ . Now, we may describe the detailed pattern  $\mathbf{T}_{Var}$  for dealing with a variable  $x$  of type Integer. It is a  $\mathcal{A}_{St}$ -weft, which is obtained by annotating  $\mathbf{S}_{Var}$ : each ingredient of  $\mathbf{S}_{Var}$  is associated to the indices of the properties which are required for its interpretation.

The *Ambi*-weft  $\mathbf{S}_{Int}$  has nothing to do with the state: we say that is *static*, which means that each point (resp. arrow, equation) of  $\mathbf{S}_{Int}$  is annotated by  $St$  (resp.  $st$ ,  $st_{\equiv}$ ). This can be extended to the constraint of  $\mathbf{S}_{Int}$ : the interpretation of  $U$  is a set  $X \times \mathbb{Q}$  such that, for each set  $Y \times \mathbb{Q}$ , there is a unique static map  $g : Y \times \mathbb{Q} \rightarrow X \times \mathbb{Q}$ ; this means that  $X$  is a one-element set, or, equivalently, that the point  $U$  is interpreted as the set  $\mathbb{Q}$ . The equation  $v \circ u \equiv id_Z$  is an observation equation: the interpretation of  $v \circ u$  is not an identity map, it only looks like an identity map upon the values, but it may modify the state.

$\mathcal{A}_{St}$ -WEFT  $\mathbf{T}_{Var}$ :

*points*:  $St$ :  $U, Z$ ,  
*arrows*:  $st$ :  $z, s, p$ ,  
 $acc$ :  $v$ ,  
 $mod$ :  $u$ ,  
*equations*:  $st_{\equiv}$ :  $p \circ s \equiv id_Z, s \circ p \equiv id_Z$ ,  
 $obs_{\equiv}$ :  $v \circ u \equiv id_Z$ ,  
*constraint*: “static”:  $U = 1$ .



Actually, in the description above, for clarity, we have only mentioned “the strongest known index”: for instance,  $\mathbf{T}_{Var}([\mathbf{Ar}, acc])$  is made of the arrows which are annotated by either *acc* or *st*. It is easy to see that, if we forget the indices, *i.e.* if we forget the middle column in this description, we get the *Ambi*-weft  $\mathbf{S}_{Var}$ . More precisely, by forgetting the indices we get a *loose* homomorphism (as defined in [guide2])  $R_{Var} : \mathbf{T}_{Var} \rightarrow \mathbf{S}_{Var} \circ \rho_{St}$  of  $\mathcal{A}_{St}$ -wefts. This is *not* a homomorphism of wefts, because the image of the constraint of  $\mathbf{T}_{Var}$  is *not* a constraint with the same shape.

So, the description of  $\mathbf{T}_{Var}$  above is, in fact, a description of the whole stratification  $(\mathbf{S}_{Var}, \mathbf{T}_{Var}, R_{Var})$ . This description is quite similar to various notions of algebraic specifications with implicit state, like those in [Gaudel *et al.* 99], [Lellahi Zamulin 99], among many others. What is really new here, is the strong structure which is given to this description: it is a stratification along  $\rho_{St}$ , and  $\mathbf{T}_{Var}$  is a weft, *i.e.* some kind of (generalized) algebraic specification. This is possible because, here, the usual “key-words” (like *static*, *access*, ...) are given a formal status: they can be identified to the ingredients of the projective sketch  $\mathbf{F}_{St}$ . A consequence is that  $\mathbf{T}_{Var}$ , as any weft, has *realizations*. Their definition can be found in [guide1], it generalizes the definition of the set-valued realizations of an *Ambi*-weft. Realizations of  $\mathbf{T}_{Var}$  will be used in section 4.2.

### 3.5 The forms of the pieces

In order to express precisely the properties which correspond to the indices, we now use other wefts. These are *Ambi*-wefts, *i.e.*, essentially, usual algebraic specifications. Since the properties involve the set  $\mathbb{Q}$  of states, we must now, but only now, introduce a point  $Q$  for specifying the set of states.

In order to take into account the set of states, we said in section 3.3 that the interpretation of each point of  $\mathbf{S}$  should be a set  $X \times \mathbb{Q}$ . It means that it should be a set-valued realization of the *Ambi*-weft  $\kappa_{St}([\mathbf{Pt}, St])$  below.

*Ambi*-WEFT  $\kappa_{St}([\mathbf{Pt}, St])$ :

points:  $H, H', E$ ,

arrows:  $h : H' \rightarrow H, h^q : H' \rightarrow E$ ,

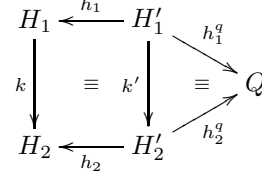
constraint:  $(H'; h, h^q)$ .

$$H \xleftarrow{h} H' \xrightarrow{h^q} Q$$

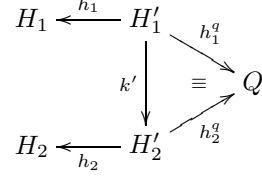
From now on, for each integer  $n$ , let  $\kappa_{St}([\mathbf{Pt}, St])_n$  denote a copy of  $\kappa_{St}([\mathbf{Pt}, St])$ , where each ingredient has suffix  $n$ , except for the point  $E$ . Hence the weft  $\kappa_{St}([\mathbf{Pt}, St])_{n_1} + \kappa_{St}([\mathbf{Pt}, St])_{n_2}$  (with  $n_1 \neq n_2$ ), which is obtained by merging  $\kappa_{St}([\mathbf{Pt}, St])_{n_1}$  and  $\kappa_{St}([\mathbf{Pt}, St])_{n_2}$ , has only one copy of the point  $E$ .

Now, an arrow is a static (*resp.* an access, a modification, an initialization) arrow if its interpretation is a set-valued realization of the weft  $\kappa_{St}([\mathbf{Ar}, st])$  (*resp.*  $\kappa_{St}([\mathbf{Ar}, acc])$ ,  $\kappa_{St}([\mathbf{Ar}, mod])$ ,  $\kappa_{St}([\mathbf{Ar}, ini])$ ) below.

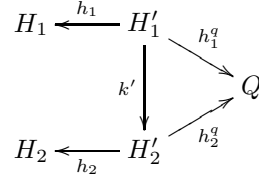
*Ambi-WEFT*  $\kappa_{St}([\mathbf{Ar}, st])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k : H_1 \rightarrow H_2, k' : H'_1 \rightarrow H'_2$ ,  
*equations:*  $h_2 \circ k' \equiv k \circ h_1, h_2^q \circ k' \equiv h_1^q$ .



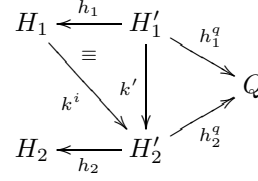
*Ambi-WEFT*  $\kappa_{St}([\mathbf{Ar}, acc])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k' : H'_1 \rightarrow H'_2$ ,  
*equations:*  $h_2^q \circ k' \equiv h_1^q$ .



*Ambi-WEFT*  $\kappa_{St}([\mathbf{Ar}, mod])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k' : H'_1 \rightarrow H'_2$ .



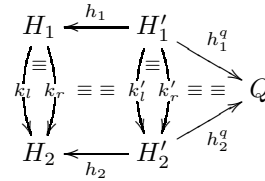
*Ambi-WEFT*  $\kappa_{St}([\mathbf{Ar}, ini])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k' : H'_1 \rightarrow H'_2, k^i : H_1 \rightarrow H'_2$ ,  
*equation:*  $k' \equiv k^i \circ h_1$ .



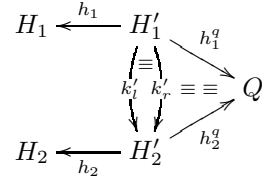
Note that  $\kappa_{St}([\mathbf{Ar}, st])$  extends  $\kappa_{St}([\mathbf{Ar}, acc])$ , which extends  $\kappa_{St}([\mathbf{Ar}, mod])$ , and that  $\kappa_{St}([\mathbf{Ar}, ini])$  also extends  $\kappa_{St}([\mathbf{Ar}, mod])$ . This corresponds to the fact that:  $st \Rightarrow acc \Rightarrow mod$  and  $ini \Rightarrow mod$ .

Similarly, an equation is a static (resp. an access, a strong, an observation) equation if its interpretation is a set-valued realization of the weft  $\kappa_{St}([\mathbf{Eq}, st_{\equiv}])$  (resp.  $\kappa_{St}([\mathbf{Eq}, acc_{\equiv}])$ ,  $\kappa_{St}([\mathbf{Eq}, stg_{\equiv}])$ ,  $\kappa_{St}([\mathbf{Eq}, obs_{\equiv}])$ ) below.

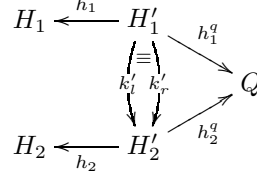
*Ambi-WEFT*  $\kappa_{St}([\mathbf{Eq}, st_{\equiv}])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1, \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k_l, k_r : H_1 \rightarrow H_2$ ,  
 $k'_l, k'_r : H'_1 \rightarrow H'_2$ ,  
*equations:*  $h_2 \circ k'_l \equiv k_l \circ h_1, h_2^q \circ k'_l \equiv h_1^q$ ,  
 $h_2 \circ k'_r \equiv k_r \circ h_1, h_2^q \circ k'_r \equiv h_1^q$ ,  
 $k_l \equiv k_r, k'_l \equiv k'_r$ .



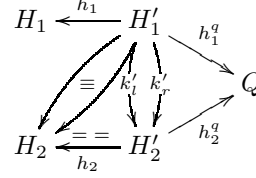
*Ambi-WEFT*  $\kappa_{St}([\mathbf{Eq}, acc_{\equiv}])$ :  
*extends:*  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows:*  $k'_l, k'_r : H'_1 \rightarrow H'_2$ ,  
*equations:*  $h_2^q \circ k'_l \equiv h_1^q, h_2^q \circ k'_r \equiv h_1^q$ ,  
 $k'_l \equiv k'_r$ .



*Ambi*-WEFTS  $\kappa_{St}([\mathbf{Eq}, stg_{\equiv}])$ :  
*extends*:  $\kappa_{St}([\mathbf{Pt}, St])_1 + \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows*:  $k'_l, k'_r : H'_1 \rightarrow H'_2$ ,  
*equations*:  $k'_l \equiv k'_r$ .



*Ambi*-WEFT  $\kappa_{St}([\mathbf{Eq}, obs_{\equiv}])$ :  
*extends*:  $\kappa_{St}([\mathbf{Pt}, St])_1, \kappa_{St}([\mathbf{Pt}, St])_2$ ,  
*arrows*:  $k'_l, k'_r : H'_1 \rightarrow H'_2$ ,  
*equation*:  $h_2 \circ k'_l \equiv h_2 \circ k'_r$ .



Note that  $\kappa_{St}([\mathbf{Eq}, st_{\equiv}])$  extends  $\kappa_{St}([\mathbf{Eq}, acc_{\equiv}])$ , which extends  $\kappa_{St}([\mathbf{Eq}, stg_{\equiv}])$ . In addition (up to an innocuous extension of  $\kappa_{St}([\mathbf{Eq}, stg_{\equiv}])$ ), the *Ambi*-weft  $\kappa_{St}([\mathbf{Eq}, stg_{\equiv}])$  extends  $\kappa_{St}([\mathbf{Eq}, obs_{\equiv}])$ . All this corresponds to the fact that:  
 $st_{\equiv} \Rightarrow acc_{\equiv} \Rightarrow stg_{\equiv} \Rightarrow obs_{\equiv}$ .

In this way, we associate to each point  $F = [E, i]$  of  $\mathbf{F}_{St}$  (where  $E$  is  $\mathbf{Pt}$ ,  $\mathbf{Ar}$  or  $\mathbf{Eq}$ ) an *Ambi*-weft  $\kappa_{St}(F)$ . We may also associate to each arrow  $f : F \rightarrow F'$  of  $\mathbf{F}_{St}$  a homomorphism of *Ambi*-wefts  $\kappa_{St}(f) : \kappa_{St}(F') \rightarrow \kappa_{St}(F)$ , up to changing its direction. For instance, the arrow  $[dom, st] : [\mathbf{Ar}, st] \rightarrow [\mathbf{Pt}, St]$  is associated to the homomorphism which sends  $\kappa_{St}([\mathbf{Pt}, St])$  onto the subweft  $\kappa_{St}([\mathbf{Pt}, St])_1$  of  $\kappa_{St}([\mathbf{Ar}, st])$ . The arrow  $[id_{\mathbf{Ar}}, st \Rightarrow acc] : [\mathbf{Ar}, st] \rightarrow [\mathbf{Ar}, acc]$  is associated to the homomorphism  $\kappa_{St}([id_{\mathbf{Ar}}, st \Rightarrow acc])$  which is the extension homomorphism from  $\kappa_{St}([\mathbf{Ar}, st])$  to  $\kappa_{St}([\mathbf{Ar}, acc])$ .

More generally,  $\kappa_{St}$  could easily be completed in order to get a counter-model of  $\mathbf{F}_{St}$  with values in  $\mathcal{Weft}(\mathbf{Ambi})$ , where a *counter-model* of a projective sketch is defined, essentially, as a realization which changes the direction of arrows.

To recap, an ingredient of  $\mathbf{S}$  of nature  $E$  and index  $i$  should be interpreted as a set-valued realization of the *Ambi*-weft  $\kappa_{St}([E, i])$ . We have defined a mosaic with implicit state for dealing with a variable of type Integer: this mosaic  $\Sigma_{Var}$  is made of  $\rho_{St}$ ,  $\kappa_{St}$  and  $R_{Var}$ .

## 4 Fundamental results

We are now able to give a strong categorical structure, respectively, to the algebraic specifications with implicit state (section 4.1), to their interpretations (section 4.2), to the corresponding explicitization process (section 4.3), and to the corresponding imperative programs (section 4.4).

### 4.1 Algebraic specifications with implicit state

It follows from the definition given in section 3.1 that a mosaic with implicit state is characterized by its stratification  $R : \mathbf{T} \rightarrow \mathbf{S} \circ \rho_{St}$ . As noticed in section 3.4, the description of  $R$  is similar to the usual description of an algebraic specification with implicit state. Hence, we get our main result.

**Fact 1.** *A specification with implicit state can be identified with a mosaic with implicit state  $\Sigma$ .*

#### 4.2 The realizations of a mosaic

In section 3.2, we saw that the realizations of  $\mathbf{S}_{Var}$  are irrelevant. The required interpretation of an ingredient of  $\mathbf{S}_{Var}$  of nature  $E$  and index  $i$  should be a set-valued realization of  $\kappa_{St}([E, i])$ . According to the general definition of the realizations of any weft, as given in [guide1], the required “interpretation of  $\mathbf{S}_{Var}$ ” is a realization of  $\mathbf{T}_{Var}$  with values in  $Real(\kappa_{St}(-), Set)$ .

More generally, by definition, the *set-valued realizations* of a mosaic  $\Sigma$  are the realizations of  $\mathbf{T}$  with values in the set-valued realizations of  $\kappa(-)$ , i.e. :  $Real(\Sigma, Set) = Real(\mathbf{T}, Real(\kappa(-), Set))$ .

**Fact 2.** *The usual interpretations of a specification with implicit state can be identified with the set-valued realizations of  $\Sigma$ .*

*Remark 1.* The part of  $\mathbf{S}$  which is made of all its static and access ingredients, is a subweft of  $\mathbf{S}$ , sometimes called the *instant part* of  $\mathbf{S}$ . It can be seen as the largest subweft of  $\mathbf{S}$  which has relevant set-valued realizations: they can be identified with the observable part of the states. For instance, the instant part of  $\mathbf{S}_{Var}$  is  $\mathbf{S}_{Var}^{inst}$ , as defined in section 3.2.

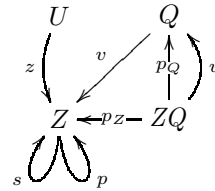
*Remark 2.* A homomorphism of wefts has a good behaviour with respect to the realizations of the wefts, whereas a loose homomorphism has a bad behaviour. Here indeed, the realizations of  $\mathbf{S}_{Var}$  are irrelevant, whereas  $\mathbf{T}_{Var}$  has relevant realizations.

#### 4.3 The explicitization of a mosaic

As explained in [guide2], we may build an *Ambi*-weft  $\mathbf{Rib}(\Sigma)$  from a mosaic  $\Sigma$  with implicit state, using the *ribbon* construction. For this purpose, we amplify each ingredient of  $\mathbf{S}$ , according to its nature and indices, thanks to the forms  $\kappa_{St}(-)$ . It means that each ingredient of  $\mathbf{S}$  is replaced by a copy of  $\kappa_{St}([E, i])$ , for each of its indices  $i$ , and these wefts are merged together “in the right way”, as defined in [guide2]. In other words, this construction follows the pattern  $\mathbf{T}$ , and replaces each element of  $\mathbf{T}([E, i])$  by a piece with form  $\kappa([E, i])$ . According to the fundamental theorem on the ribbon product, which is proven in [guide2], the realizations of a mosaic  $\Sigma$  can be identified with the realizations of the weft  $\mathbf{Rib}(\Sigma)$ , i.e. :  $Real(\mathbf{Rib}(\Sigma), Set) \cong Real(\Sigma, Set)$ .

We may easily check this result on our example, by building the *Ambi*-weft  $\mathbf{Rib}(\Sigma_{Var})$ : indeed, it is equivalent to the following *Ambi*-weft.

*Ambi*-WEFT  $\mathbf{S}_{Var}^{expl}$ :  
*extends:*  $\mathbf{S}_{Int}$ ,  
*points:*  $Q, ZQ$ ,  
*arrows:*  $p_Z : ZQ \rightarrow Z, p_Q : ZQ \rightarrow Q$ ,  
 $v : Q \rightarrow Z, u : ZQ \rightarrow Q$ ,  
*equation:*  $v \circ u \equiv p_Z : ZQ \rightarrow Z$ ,  
*constraint:*  $(ZQ; p_Z, p_Q)$ .



The required set-valued realization of  $\Sigma_{Var}$  corresponds to the following set-valued realization  $\omega_{Var}^{expl}$  of  $\mathbf{S}_{Var}^{expl}$ . It extends  $\omega_{Int}$ , and interprets the points  $Q$  and  $ZQ$  as the set  $\mathbb{Q}$  of states and the cartesian product  $\mathbb{Z} \times \mathbb{Q}$ , the arrows  $p_Z$  and  $p_Q$  as the canonical projections from  $\mathbb{Z} \times \mathbb{Q}$  towards  $\mathbb{Z}$  and  $\mathbb{Q}$ , and the arrows  $v$  and  $u$  as the maps *value* and *update* (as defined in section 3.2).

**Fact 3.** *The explicitization of a specification with implicit state can be identified with the *Ambi*-weft  $\mathbf{Rib}(\Sigma)$ .*

*Hence, the usual interpretations of a specification with implicit state can be identified with the set-valued realizations of the *Ambi*-weft  $\mathbf{Rib}(\Sigma)$ .*

#### 4.4 The terms of a mosaic

As soon as  $\mathbf{E} = \mathbf{E}_{Ambi}$ , the *terms* of a mosaic  $\Sigma$  are defined as the terms of the *Ambi*-weft  $\mathbf{S}$ , see [guide3]. We noticed in section 3.2 that the terms of the *Ambi*-weft  $\mathbf{S}_{Var}$  correspond to imperative programs.

**Fact 4.** *The imperative programs for a specification with implicit state can be identified with the terms of  $\Sigma$ , i.e. the terms of the *Ambi*-weft  $\mathbf{S}$ .*

*Remark 1.* With our point of view, we naturally avoid the usual problem of the duplication of the state. Indeed, usually, once there is a point  $Q$  in a specification, it is difficult to avoid the point  $Q \times Q$  when building and evaluating its terms. This is true for *Ambi*-wefts as well as for algebraic specifications. However, this is false for the point  $Q$  in an *Ambi*-weft  $\mathbf{Rib}(\Sigma)$  where  $\Sigma$  is a mosaic with implicit state. Indeed, terms of  $\Sigma$  are terms of  $\mathbf{S}$ , hence when dealing with them, only *one* copy of the point  $Q$  appears in the ribbon construction, by the mere definition of this construction and the description of the *Ambi*-wefts  $\kappa_{St}(-)$ .

*Remark 2.* Clearly, computations cannot be performed in  $\mathbf{S}$ , since a non-trivial instruction may correspond to a trivial term (an example has been given in section 3.1). However, computations can be performed in  $\mathbf{T}$ , since the set-valued realizations of  $\Sigma$  are realizations of  $\mathbf{T}$ . Hence, we are able to prove that *computations may be performed without knowing  $\kappa$ , i.e. without using the point  $Q$ .*

## 5 Conclusion

In this paper, we have given a strong categorical structure to the algebraic specifications with implicit state, by identifying them to mosaics with implicit state. For this purpose, we have introduced wefts, which generalize algebraic specifications.

Since the forms of the pieces are the same for all mosaics with implicit state, such a mosaic is characterized by its patterns  $\mathbf{S}$  and  $\mathbf{T}$  (and the relation  $R$  between them). The approximate pattern  $\mathbf{S}$  is an *Ambi*-weft, it can be identified with an algebraic specification, in the usual sense. On the contrary, the detailed pattern  $\mathbf{T}$  is an  $\mathcal{A}_{St}$ -weft, it *cannot* be identified with an algebraic specification. However, whereas  $\mathbf{S}$  is sufficient for writing the imperative programs,  $\mathbf{T}$  is needed for understanding the meaning of the mosaic, and for building the framework for the evaluation of the imperative programs. This is why we need a general

definition of  $\mathcal{A}$ -wefts (for any category  $\mathcal{A}$ ), in order to give a coherent definition of mosaics with implicit state.

In this short paper, we have only considered a very simple example, however the study of more sophisticated examples would be fairly similar. The study of imperative programs and their evaluation would deserve a detailed study, in a subsequent paper, relying on [guide3].

In addition, other mosaics (with different  $\rho$  and  $\kappa$ ) can be used for dealing with various implicit features in computer science. An example of error handling can be found in [guide2], other applications are in progress.

**Acknowledgments.** We warmly thank the participants of the working group *sketches and computer algebra*, mainly Kazem Lellahi, Catherine Oriat and Jean-Claude Reynaud.

## References

- [ref1] C. Lair and D. Duval. Sketches and specifications: Reference manual. First part: Compositive graphs. *Rapport de recherche LACO*, 2000, <http://www.unilim.fr/laco/rapports>.
- [ref2] C. Lair and D. Duval. Sketches and specifications: Reference manual. Second part: Projective sketches. *Rapport de recherche LACO*, 2000, <http://www.unilim.fr/laco/rapports>.
- [guide1] D. Duval and C. Lair. Sketches and specifications: User's guide. First part: Wefts for explicit specification. *Rapport de recherche LACO*, 2000, <http://www.unilim.fr/laco/rapports>. Submitted for publication.
- [guide2] D. Duval and C. Lair. Sketches and specifications: User's guide. Second part: Mosaics for implicit specification. *Rapport de recherche LACO*, 2000, <http://www.unilim.fr/laco/rapports>. Submitted for publication.
- [guide3] D. Duval and C. Lair. Sketches and specifications: User's guide. Third part: Functional and imperative programs. Work in progress.
- [Astesiano et al. 99] E. Astesiano, H.-J. Kreowski and B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. Springer, 1999.
- [Dauchy Gaudel 94] P. Dauchy and M.-C. Gaudel. Algebraic specifications with implicit state. Rapport de Recherche 887, Univ. de Paris-Sud, Laboratoire de Recherche en Informatique, 1994.
- [Ehresmann 66] C. Ehresmann. Introduction to the theory of structured categories. Technical Report 10, Univ. of Kansas at Lawrence, 1966.
- [Ehresmann 68] C. Ehresmann. Esquisses et types de structures algébriques. *Bulletin de l'Institut Polytechnique, Iași*, 14:1–32, 1968.
- [Gaudel et al. 99] M.-C. Gaudel, C. Houry and A. Zamulin. Dynamic Systems with Implicit State. In *ETAPS'99 proceedings*, LNCS 1577, pages 114-128. Springer-Verlag, 1999.
- [Lair 87] C. Lair. Trames et sémantiques catégoriques des systèmes de trames. *Diagrammes*, 18:CL1–CL47, 1987.
- [Lellahi Zamulin 99] K. Lellahi and A. Zamulin. Dynamic systems based on update sets. Rapport de recherche 99-03, Univ. de Paris-Nord, Institut Galilée, 1999. Submitted for publication.