REPORTS
IN
INFORMATICS

ISSN 0333-3590

From Indexed to Fibred Semantics
— The Generalized Sketch File —

Uwe Wolter, Department of Informatics,
University of Bergen, Norway
Zinovy Diskin, Department of Computer Science, University of Toronto, Canada

REPORT NO 361 **October 2007**

Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/pdf/2007-361.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Contents

1 Introduction **1**

2 Graphs and Van Kampen Squares **2**

3 Generalized Sketches **4**

4 Indexed Semantics of Sketches **9**

5 Sketch Transformations and Sketch Operations **10**

6 Grothendieck Construction **13**

7 From Indexed to Fibred Semantics **18**

8 Concluding Discussion and Open Questions **21**

Abstract

Generalized sketches is a graph-based specification framework that borrows its main ideas from both categorical and first-order logic, and adapts them to software engineering needs. As a new result concerning the syntax of this framework we prove in this paper that categories of sketches are as well finitely complete as finitely cocomplete. Concerning the semantics we show, first, that the indexed semantics of Generalized Sketches offers "amalgamated sums" of models and a corresponding "extension lemma". Next, we investigate how the indexed semantics can be transformed into a fibred semantics by means of (a variant of) the Grothendieck construction. Analyzing what happans with the amalgamation of models under this transformation we observe that the Grothendieck construction transforms pushouts into "weak" van Kampen squares. [1] [2]

1 Introduction

In this paper we continue our research on theory and application of the Generalized Sketch (GS) framework. The machinery of generalized sketches was developed and applied in a few industrial projects in Latvia in 1993-94 and the corresponding logic was presented at Logic Colloquium'95 [Dis97]. Even earlier, Michael Makkai came to the need to generalize the notion of Ehresmann's sketches from his work on an abstract formulation of Completeness Theorems in logic [Mak97]. We refer to the historical remarks section in [DW07] for more details.

The main goal of the present paper is to bridge the gap between the "indexed version" of the GS framework presented in [WD07] and the practice oriented "firbred version" in [DW07]. In addition, we rearrange the main syntactic concepts like (generalized) sketch, sketch morphism and sketch transformation as natural extensions of the concepts of graph, graph homomorphism/ production and graph transformation, respectively. As a new result we show that categories of sketches are finitely complete and finitely cocomplete.

Yet the main emphasis is on semantics. We present the indexed semantics of sketches, i.e., the classical concept of a sketch model in the GS setting [Dis96, WD07], and as new results we show the existence of "amalgamated sums" of models and (the first part of) a corresponding "extension lemma" (compare [EM85]).

For the algebraic specification community it is quite natural to describe semantics in an "indexed manner", i.e., as an interpretation of syntactic entities in a semantic universe with usually infinite semantic objects. Variations of this indexed viewpoint can be found in Universal Algebra, Algebraic Specifications, Functorial Semantics, Denotational Semantics, Institutions, and at many other places. Following this "cultural guideline" the original definitions of the semantics of generalized sketches have been based on indexed concepts too.

"Indexed theorists" have to face, however, the painful truth that "practitioners" in programming and software engineering prefer to think in terms of typed/fibred concepts. And it seem that we can do nothing about it since this preference is just reflecting the "real world" the "practitioners" are living in. Programming is dealt with a finite bunch of typed data allocations and software design, nowadays, is about drawing and (hopefully) understanding different kinds of finite diagrams with different types of nodes and arrows.

[1] Research partially supported by the Norwegian NFR project SHIP/VERDIKT.

[2] I'm very grateful to José Fiadeiro and Reiko Heckel for inviting me to Leicester in September 2007 to give a talk on Generalized Sketches. In the talk, I have specially addressed the transition between indexed and fibred concepts. The present paper has been triggered by that talk and the intense discussions on the next day after it.

To bridge this "cultural gap" we have to transfer our concepts and results from the indexed world into the fibred world. Such a transformation will be based on (different variants of) the so-called Grothendieck construction [BW90, MWH07]. Since the GS framework is based on graphs, we will present here a variant of the Grothendieck construction for graphs and we will show how the indexed definitions can be transformed into corresponding fibred definitions along this construction.

It is folklore that the Grothendieck construction turns composition, i.e., commutative triangles, into pullback diagrams. Analyzing how amalgamation of models is transformed by the Grothendieck construction, we have found that

the Grothendieck construction transforms pushouts into "weak" van Kampen squares.

As far as we know, this insight has yet not been formulated in the literature.

Van Kampen squares are the essential ingredient in the definition of Adhesive Categories as they have been introduced recently by Lack and Sobociński [LS04], and adhesive categories are considered as a promising formal basis for a general theory of transformations and of reactive systems. For example, [EEPT06] provides a thorough revision and generalization of the different variants of graph-transformations and high level replacement systems based on the concept of adhesive category.

That the GS framework is well-suited and appropriate for defining the syntax and semantics of a wide class of diagrammatic specification techniques, as ER diagrams, database schemata, UML class diagrams, has been shown in a sequence of papers [DK97, DKPJ00, Dis03, Dis05]. Therefore we will concentrate here on the discussion of how the Algebraic Specification formalism can be reflected within the GS framework. In such a way we will hopefully make apparent the potentials of the GS framework and the place of this framework within the landscape of specification formalisms.

2 Graphs and Van Kampen Squares

The Generalized Sketch framework, as any other diagrammatic specification technique, is heavily based on the concept of graph, thus we start with a formal definition of this concept and other necessary concepts and results around. Before we dive into technicalities, the following important remark is in order. In this paper we are concerned with (at least) two different kinds of "diagrams". To minimize potential confusion, we will try to distinguish between them with the following terminology.

1. There is the general idea of a "picture" with (labeled) nodes and (labeled) edges. In software engineering we meet different specializations of this general idea for specification purposes like "ER diagrams", "UML class diagrams", ... We will, therefore, refer to those specializations as **model diagrams** or **mod-diagrams**.

2. Finally, we have a strict, formal mathematical definition of diagrams. We will call these diagrams **math-diagrams** or simple **diagrams**.

We adapt the notation from [Fia05].

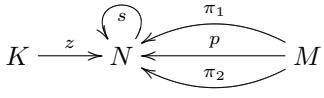
Definition 1 (Graph)**.** *A **graph** $G = (G_0, G_1, sc, tg)$ is given by*

- *a collection G_0 of nodes,*
- *a collection G_1 of arrows,*
- *a map $sc : G_1 \to G_0$ assigning to each arrow its **source**, and*
- *a map $tg : G_1 \to G_0$ assigning to each arrow its **target**.*

We usually write $f : x \to y$ or $x \xrightarrow{f} y$ to indicate that $sc(f) = x$ and $tg(f) = y$.

□

Example 1 (Finite Graphs)**.** *Finite graphs are often visualized by mod-diagrams. The graph G with $G_0 = \{K, N, M\}$, $G_1 = \{z, s, p, \pi_1, \pi_2\}$ and $sc(z) = K$, $tg(z) = sc(s) = tg(s) = N$, $sc(\pi_1) = sc(\pi_2) = sc(p) = M$, and $tg(\pi_1) = tg(\pi_2) = tg(p) = N$, for example, can be visualized by the mod-diagram*



Definition 2 (Graph Homomorphism)**.** *A **graph homomorphims** $\varphi : G \to H$ is a pair of maps $\varphi_0 : G_0 \to H_0$ and $\varphi_1 : G_1 \to H_1$ such that for each arrow $f : x \to y$ of G we have $\varphi_1(f) : \varphi_0(x) \to \varphi_0(y)$ in H, i.e., we have $sr^H(\varphi_1(f)) = \varphi_0(sr^G(f))$ and $tg^H(\varphi_1(f)) = \varphi_0(tg^G(f))$ for all $f \in G_1$.*

*The **composition** $\varphi;\psi : G \to K$ of two graph homomorphisms $\varphi : G \to H$ and $\psi : H \to K$ is defined component-wise*

$$\varphi;\psi = (\varphi_0, \varphi_1);(\psi_0, \psi_1) \stackrel{def}{=} (\varphi_0;\psi_0, \varphi_1;\psi_1).$$

We will omit subscripts if convenient and no confusion arises.

Fact 1 (Category of Graphs)**.** *It is immediate to see that the composition $\varphi;\psi : G \to K$ is indeed a graph homomorphism. Identical graph homomorphisms $id_G : G \to G$ are defined by $id_G = (id_{G_0}, id_{G_1})$, and the component-wise definition of identities and composition ensures that we inherit associativity and neutrality from the category* Set*. In such a way, we obtain two categories*

- *The category* Graph *of **small graphs**, i.e., of graphs with sets of nodes and sets of arrows, and all graph homomorphisms between them.*
- *The category* GRAPH *of arbitrary graphs and all graph homomorphisms between them.*

Remark 1 (Other Categories)**.** *Beside the categories* Graph *and* GRAPH *we will also refer to other categories.*

By Set *we denote the category with objects all sets A and arrows all (total) maps $f : A \to B$. For any set A there is an **identical map** $id_A : A \to A$ and for any two maps $f : A \to B$, $g : B \to C$ there is the **composition** $f;g : A \to C$ defined by $f;g(a) = g(f(a))$ for all $a \in A$. Composition is associative and the identical maps are neutral w.r.t. composition.*

By Par *we denote the category with objects all sets A and with arrows all **partial maps** $f : A \rightharpoonup B$ where we denote the **domain of definition** of f by $dom(f) \subseteq A$. The identities $id_A : A \to A$ are the total identical maps and the composition $f;g : A \rightharpoonup C$ of two of two partial maps $f : A \rightharpoonup B$ and $g : B \rightharpoonup C$ is defined by*

$$dom(f;g) \stackrel{def}{=} \{a \in A \mid a \in dom(f), f(a) \in dom(g)\} \quad \text{and} \quad f;g(a) \stackrel{def}{=} g(f(a)) \text{ for all } a \in dom(f;g)$$

Another useful category is the category Pow *of "multi-functions" with objects all sets A and with arrows all maps $f : A \to \mathcal{P}(B)$ where $\mathcal{P}(B)$ is the **power set** of B, i.e., the set of all subsets of B. The identities $id_A : A \to \mathcal{P}(A)$ are defined by $id_A(a) = \{a\}$ for all $a \in A$ and the composition $f;g : A \to \mathcal{P}(C)$ of two maps $f : A \to \mathcal{P}(B)$ and $g : B \to \mathcal{P}(C)$ is defined by*

$$f;g(a) = g(f(a)) \stackrel{def}{=} \bigcup\{g(b) \mid b \in f(a)\} \quad \text{for all } a \in A$$

□

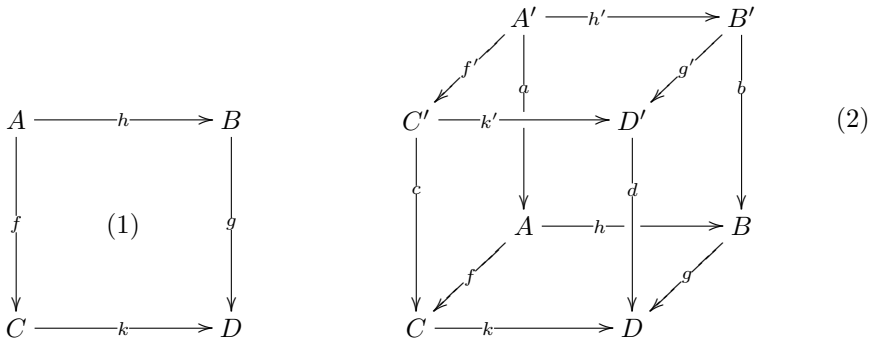
Transformations can be usually described by pushout constructions in corresponding appropriate categories. And fortunately pushouts (and pullbacks) in Graph can be constructed easily (see [EEPT06] for details).

Fact 2 (Pushouts and Pullbacks)**.** ***Pullbacks** and **pushouts** in* Graph *can be constructed componentwise for nodes and edges in* Set *and the **monomorphisms** in* Graph *are exactly the injective graph morphisms.*

To have a reasonable transformation formalism we need a certain compatiblity between pullbacks and pushouts as it can be formulated by the so-called van Kampen (VK) square. The idea is that a pushout is stable under pullbacks, and, vice versa, that pullbacks are stable under combined pushouts and pullbacks. The name "van Kampen" is due to the relationship between these squares and the van Kampen Theorem in topology. For later discussions in this paper we also coin the concept a "semi VK square".

Definition 3 (Van Kampen Square)**.** *A pushout (1) is a **van Kampen square** if, for any commutative cube (2) with (1) in the bottom and where the back face and the left face are pullbacks, the following equivalence holds:*

The top face is a pushout iff the front face and the right face are pullbacks:



*The pushout (1) is a **semi van Kampen square** if, for any commutative cube (2) with (1) in the bottom and where the back face and the left face are pullbacks, the following implication holds:*
The top face is a pushout if the front face and the right face are pullbacks:

In most categories pullbacks are not stable under arbitrary combined pushouts and pullbacks, thus in most categories not all pushouts are VK squares. In Set and Graph, however, pushouts along monomorphisms are VK squares. This fact leads to the axiomatization of adhesive categories as given in

Definition 4 (Adhesive Category)**.** *A category* $\mathbf{C}$ *is an **adhesive category** if:*

1. $\mathbf{C}$ *has pushouts along monomorphisms.*
2. $\mathbf{C}$ *has pullbacks.*
3. *Pushouts along monomorphisms are VK squares.*

On the other side, arbitray pushouts are stable under pullbacks in Set and thus also in Graph due to Fact 2.

Fact 3 (Semi VK Squares)**.** *All pushouts in* Set *and in* Graph *are semi VK squares.*

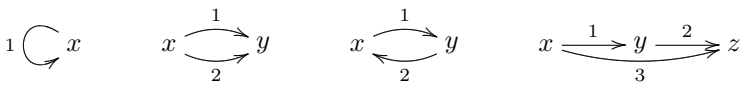
3 Generalized Sketches

Sketches are the specification formalism offered by Category Theory (CT). It was originated by Charles Ehresmann in the 60s, who invented the so called *sketches* (see [Wel] for a survey). Ehresmann's sketches were promoted for applications in computer science by Michael Barr and Charles Wells [BW90] and applied to data modeling problems by Michael Johnson and Roberth Rosebrugh [JRW02]. From now on in the paper, the term sketch will refer to a generalized sketch.

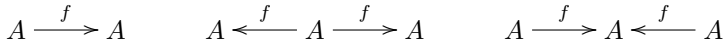
A sketch is a graph together with a set of designated diagrams in this graph.

Definition 5 (Diagram)**.** *Let* G *and* I *be graphs, A **diagram** in* G *with **shape*** I *is a graph homomorphism* $\delta : I \to G$*.*

Example 2 (Shapes)**.** *Some of the most simple shapes, that are used in nearly any application, are* $\texttt{Node} = (\ x\ )$*,* $\texttt{Arrow} = (x \xrightarrow{1} y)$*,* $\texttt{Span} = (x \xleftarrow{1} z \xrightarrow{2} y)$*,* $\texttt{Cospan} = (x \xrightarrow{1} z \xleftarrow{2} y)$*, and the following four graphs* `Loop`*,* `Cell`*,* `Circle`*, and* `Triangle`*, respectively:*



Remark 2 (Visualization of Diagrams)**.** *Mod-diagrams are traditionally also used to present diagrams* $\delta : I \to G$*. The essential idea is to draw a picture with a separate "place holder node" for each element in* I_0 *and with a separate "place holder arrow" for each element in* I_1*. And then we put into the "place holders" the corresponding items from* G *according to the assignment* δ*. In such a way, we may have in the picture different copies of the same item from* G *at different places. The following three mod-diagrams, for example,*



represent diagrams $\delta_1 : \texttt{Arrow} \to \mathsf{Set}$, $\delta_2 : \texttt{Span} \to \mathsf{Set}$, $\delta_3 : \texttt{Cospan} \to \mathsf{Set}$ *with the same "actual values" but with different shapes.*

In the classical sketch framework we have three "pre-defined" types of diagrams: Commutative, limit, and colimit diagrams. The idea of generalized sketches is to allow instead, of these three pre-defined types of diagrams, arbitrary "user-defined" types of diagrams that are appropriate for the corresponding application domain.

Definition 6 (Signature)**.** *A* ***signature*** $\Theta = (\Pi, \alpha)$ *is given by*

- *a collection* Π *of* ***(predicate) labels or symbols*** *and*
- *a function* $\alpha : \Pi \to \mathsf{Graph}_0$ *assigning to each label* $P \in \Pi$ *its* ***arity (shape)*** $\alpha(P)$.

Remark 3 (Compound Labels)**.** *In many applications it may be more "user friendly" to allow to assign to a predicate label* P *a set of possible arity shapes. We could, for example, have a label* $[prod]$ *with different shapes for empty, binary, ternary, ... products, respectively. We decided* ***not*** *to allow for those sets of arity shapes in the actual definition of signatures.*

In examples and applications a "user" is, of course, free to define a "generic predicate label" P *with a corresponding set of arity shapes. But, this will be interpreted as a "user defined mechanism" to create* ***compound labels****. In case of the above mentioned generic label* $[prod]$ *we could use, for example, the compound labels* $([prod], 0)$, $([prod], 2)$, $([prod], 3)$, ...

Remark 4 (Extensions of the Concept of Signatures)**.** *Makkai's original definition in [Mak97] is more general in the sense that he allows for arbitrary base categories* Base *instead of* Graph *only. Especially he is concerned about hierachies of pre-sheaf topoi, starting with the pre-sheaf topos* Graph*, where the category of (multi) sketches of one level is used as the base category for the next level.*

Dependencies between predicates turn out to be essential in applications of generalized sketches, but they have not been considered neither by Makkai nor by Diskin in the original definitions. To treat dependencies between single predicates we extended therefore recently the definition of signatures by considering Π *to be a category and* α *to be a functor (see [DW07, WD07]). This extension, however, will be not of interest here.*

Specifications within the Generalized Sketch framework are meant to be

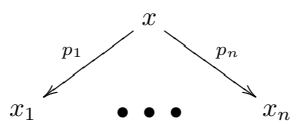
Definition 7 ((Generalized) Sketches)**.** *Given a signature* $\Theta = (\Pi, \alpha)$ *a* Θ***-sketch*** $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ *consists of a graph* $G^{\mathcal{S}}$ *and a* Π*-indexed family* $\mathcal{S}(\Pi) = (\mathcal{S}(P) \mid P \in \Pi)$ *of sets of* ***marked diagrams****, i.e., for every label* $P \in \Pi$ *there is a (maybe, empty) set* $\mathcal{S}(P)$ *of diagrams* $(P, \delta : \alpha(P) \to G^{\mathcal{S}})$ *in* $G^{\mathcal{S}}$ ***marked by*** P.

Now we want to look at how basic traditional specification techniques, as Equational Algebraic Specifications, can be reflected by diagram signatures and sketches.

Example 3 (Algebraic Signatures)**.** *Algebraic signatures* $\Sigma = (S, OP)$ *declare a set* S *of sort symbols and a set* OP *of operation symbols together with corresponding arity requirements* $op : s_1 \dots s_n \to s$*. The only "semantic" requirement in this "specification formalism" is that for any* Σ*-algebra* $\mathcal{A}$ *the sequence* $s_1 \dots s_n$ *of sort symbols has to be interpreted by the cartesian product* $\mathcal{A}(s_1) \times \dots \times \mathcal{A}(s_n)$ *of the interpretations of the corresponding single sort symbols. Therefore, the sketching of the formalism "algebraic signatures" may lead us to a diagram signature* $\Theta_{\mathbf{AS}} = (\Pi_{\mathbf{AS}}, \alpha_{\mathbf{AS}})$ *that defines infinite many labels*

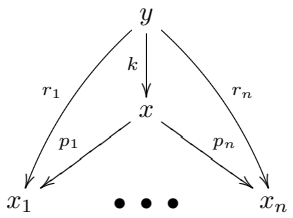
$$\Pi_{\mathbf{AS}} = \{([prod], 0), ([prod], 2), ([prod], 3), \dots, ([prod], n), \dots\}$$

with arities $\alpha_{\mathbf{AS}}([prod], 0) = \texttt{Node}$, $\alpha_{\mathbf{AS}}([prod], 2) = \texttt{Span}$ *and for any* $n \in \mathbb{N}$ *with* $n > 2$ *the arity* $\alpha_{\mathbf{AS}}([prod], n)$ *will be given by the following graph* $\texttt{Span}_n$



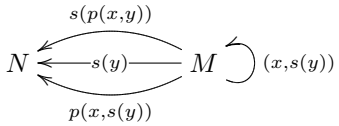
A $\Theta_{\mathbf{AS}}$*-sketch* $\mathcal{S}_{\Sigma_{Nat}}$ *that sketches the algebraic signature* Σ_{Nat} *with* $S_{Nat} = \{N\}$ *and* $OP_{Nat} = \{z : \to N, s : N \to N, p : N\,N \to N\}$*, for example, is given by the graph* G *in Example 1 together with the two marked diagrams* $\mathcal{S}_{\Sigma_{Nat}}([prod], 0) = \{K\}$, $\mathcal{S}_{\Sigma_{Nat}}([prod], 2) = \{N \xleftarrow{\pi_1} M \xrightarrow{\pi_2} N\}$*. All the other sets* $\mathcal{S}_{\Sigma_{Nat}}([prod], n)$ *will be empty for* $n > 2$*. Note, how the implicit assumption in* Σ_{Nat} *that the sequence* $N\,N$ *of sort symbols has to be interpreted by a product has been made explicit in the sketch* $\mathcal{S}_{\Sigma_{Nat}}$.

Example 4 (Equational Specifications)**.** *An **equational specification** $SP = (\Sigma, EQ)$ is given by an algebraic signature $\Sigma = (S, OP)$ and a set EQ of Σ-equations $(t_1 = t_2)$ with t_1 and t_2 Σ-terms of the same sort. To reflect the concept of terms the intended diagram signature $\Theta_{\mathbf{EQ}}$ has to include, besides the "product labels" $([prod], n)$ from $\Pi_{\mathbf{AS}}$, additional labels: A reasonable choice could be a label $[comp]$ with arity* `Triangle` *to reflect the compostion of maps, a label $[=]$ with arity* `Cell` *to write equations, and compound labels $([tupl], n)$, $n \geq 2$ with arities given by the following diagrams* $\texttt{Tupl}_n$

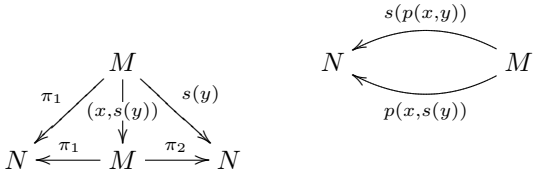


to reflect the tupling of terms. To describe equations with a multiple occurence of a single variable we introduce further a label $[id]$ with arity `Loop` *for indicating identical maps.*

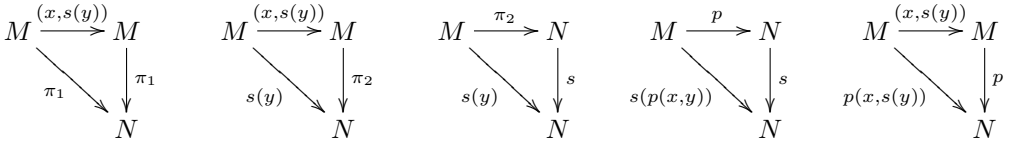
A $\Theta_{\mathbf{EQ}}$-sketch $\mathcal{S}_{SP_{Nat}}$ that correponds to the equational specification $SP_{Nat} = (\Sigma_{Nat}, EQ_{Nat})$ with a single equation $EQ_{Nat} = \{(p(x, s(y)) = s(p(x, y))\}$, for example, will be given by the graph G in Example 1 extended by the following four arrows



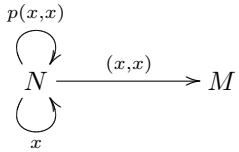
There are no additional diagrams marked by product labels, i.e., we have $\mathcal{S}_{SP_{Nat}}([prod], n) = \mathcal{S}_{\Sigma_{Nat}}([prod], n)$, for $n = 0, 2, 3, \ldots$. The sets $\mathcal{S}_{SP_{Nat}}([tupl], 2)$ and $\mathcal{S}_{SP_{Nat}}([=])$ are singleton sets visualized by the following mod-diagrams



$\mathcal{S}_{SP_{Nat}}([comp])$ contains five diagrams. The first two diagrams arise from tupling and the other three from composition.

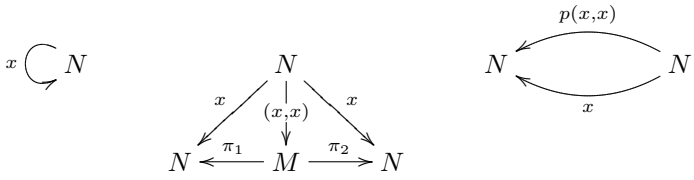


Another $\Theta_{\mathbf{EQ}}$-sketch $\mathcal{S}_{SP'_{Nat}}$ that correponds to the equational specification $SP'_{Nat} = (\Sigma_{Nat}, EQ'_{Nat})$ with a single equation $EQ'_{Nat} = \{(x = p(x, x))\}$ is given by the graph G in Example 1 extended by the following three arrows

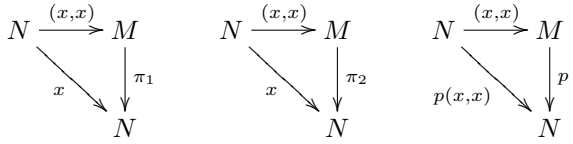


There are no additional diagrams marked by product labels, i.e., we have $\mathcal{S}_{SP_{Nat}}([prod], n) = \mathcal{S}_{\Sigma_{Nat}}([prod], n)$, for $n = 0, 2, 3, \ldots$. The sets $\mathcal{S}_{SP_{Nat}}([id], 2)$, $\mathcal{S}_{SP_{Nat}}([tupl], 2)$ and $\mathcal{S}_{SP_{Nat}}([=])$ are singleton sets visualized by

the following mod-diagrams



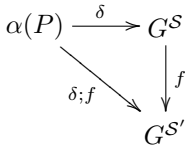
$\mathcal{S}_{SP_{Nat}}([comp])$ *contains three diagrams. The first two diagrams arise from tupling and the third from composition.*



Note, that the $\Theta_{\mathbf{EQ}}$*-sketches* $\mathcal{S}_{SP_{Nat}}$ *and* $\mathcal{S}_{SP'_{Nat}}$ *have to present explicitly all the subterms in the equations* $(p(x, s(y)) = s(p(x, y))$ *and* $(x = p(x, x))$*, respectively. In traditional equational specifications this is not necessary since the syntactical appearance of a term allows to reconstruct the inductive process of constructing the term. This feature can be seen, somehow, as the essence of the concept of term. At the present stage the Generalized Sketch framework has not incorporated such advanced possibilities to create syntactical denotations (see also the discussion in Remark 7).*

The concept of graph homomorphisms exends naturally to sketches.

Definition 8 (Sketch Morphisms)**.** *A* Θ***-sketch morphism*** $f : \mathcal{S} \to \mathcal{S}'$ *between two* Θ*-sketches* $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ *and* $\mathcal{S}' = (G^{\mathcal{S}'}, \mathcal{S}'(\Pi))$ *is a graph homomorphism* $f : G^{\mathcal{S}} \to G^{\mathcal{S}'}$ *compatible with marked diagrams, i.e.,* $(P, \delta : \alpha(P) \to G^{\mathcal{S}}) \in \mathcal{S}(P)$ *implies* $(P, \delta; f : \alpha(P) \to G^{\mathcal{S}'}) \in \mathcal{S}'(P)$ *for all* $P \in \Pi$*.*



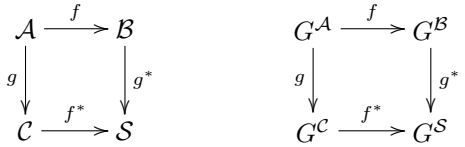
Fact 4 (Category of Sketches)**.** *The associativity of the composition of graph homomorphisms ensures that the composition of two* Θ*-sketch morphisms becomes a* Θ*-sketch morphism as well, and that the composition of* Θ*-sketch morphisms is associative too. Further the identical graph homomorphism* $id_{G^{\mathcal{S}}} = (id_{G_0^{\mathcal{S}}}, id_{G_1^{\mathcal{S}}}) : G^{\mathcal{S}} \to G^{\mathcal{S}}$ *define an identical* Θ*-sketch morphism* $id_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}$ *that is neutral w.r.t. the composition of* Θ*-sketch morphisms. In such a way, we obtain for any signature* Θ *a category* $\mathbf{Ske}(\Theta)$*.*

Example 5 (Signature and Specification Morphisms)**.** $\Theta_{\mathbf{AS}}$*-sketch morphisms represent the traditional signature morphisms [EM85, Rei87, Wol90] but allow also to map single sort symbols to sequences of sort symbols (compare the discussion in Example 9).*

$\Theta_{\mathbf{EQ}}$*-sketch morphisms represent the corresponding concept of strong specification morphisms where the equations from one specification have to be translated directly to equations in the other specification and not into derived equations.*

It is straightforward to show that the construction of pullbacks and pushouts in Graph can be extended to a construction of pullbacks and pushouts, respectively, in $\mathbf{Ske}(\Theta)$. It seems, however, to be worth to formulate these statements explicitly (for the first time).

Proposition 1 (Pushouts of Sketches)**.** *The* ***pushout*** $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ *of a span* $\mathcal{C} \xleftarrow{g} \mathcal{A} \xrightarrow{f} \mathcal{B}$ *of* Θ*-sketch morphisms is obtained by constructing the pushout in* Graph *for the underlying graph homomorphisms*

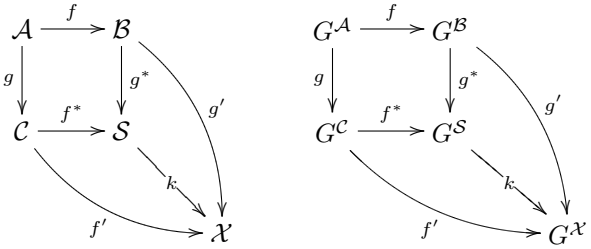


and by defining

$$\mathcal{S}(P) \stackrel{def}{=} \{(P,\delta;f^*) \mid (P,\delta) \in \mathcal{C}(P)\} \cup \{(P,\sigma;g^*) \mid (P,\sigma) \in \mathcal{B}(P)\} \quad \textit{for all} \quad P \in \Pi.$$

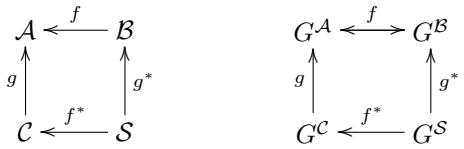
Proof. The definition of $\mathcal{S}(\Pi)$ ensures that the graph homomorphisms $f^* : G^{\mathcal{C}} \to G^{\mathcal{S}}$, $g^* : G^{\mathcal{B}} \to G^{\mathcal{S}}$ establish indeed Θ-sketch morphisms $f^* : \mathcal{C} \to \mathcal{S}$ and $g^* : \mathcal{B} \to \mathcal{S}$, respectively.

Further, there exists for all Θ-sketch morphisms $f' : \mathcal{C} \to \mathcal{X}$ and $g' : \mathcal{B} \to \mathcal{X}$ with $g; f' = f; g'$ a unique graph homomorphism $k : G^{\mathcal{S}} \to G^{\mathcal{X}}$ with $f^*; k = f'$ and $g^*; k = g'$.



For any $P \in \Pi$ and any $(P,\delta;f^*) \in \mathcal{S}(P)$ with $(P,\delta) \in \mathcal{C}(P)$ this entails $(P,\delta;f^*;k) = (P,\delta;f') \in \mathcal{X}(P)$ since f' is a Θ-sketch morphism. Analogously, we obtain $(P,\sigma;g^*;k) = (P,\delta;g') \in \mathcal{X}(P)$ for any $(P,\sigma;g^*) \in \mathcal{S}(P)$ with $(P,\sigma) \in \mathcal{B}(P)$ since g' is a Θ-sketch morphism. This proves, finally, that k defines a Θ-sketch morphism $k : \mathcal{S} \to \mathcal{X}$. □

Proposition 2 (Pullbacks of Sketches)**.** *The* ***pullback*** $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ *of a co-span* $\mathcal{C} \stackrel{g}{\to} \mathcal{A} \stackrel{f}{\leftarrow} \mathcal{B}$ *of* Θ*-sketch morphisms is obtained by constructing the pullback in* Graph *for the underlying graph homomorphisms*

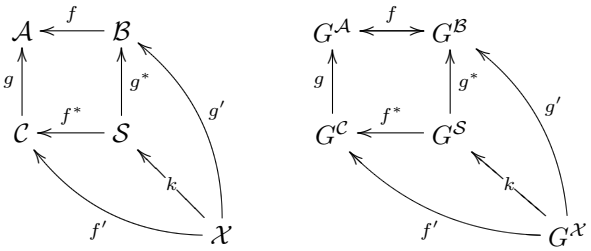


and by defining for all $P \in \Pi$*:*

$$(P,\delta) \in \mathcal{S}(P) \quad \textbf{iff} \quad (P,\delta;g^*) \in \mathcal{B}(P) \textit{ and } (P,\delta;f^*) \in \mathcal{C}(P) \textit{ and } (P,\delta;g^*;f) = (P,\delta;f^*;g) \in \mathcal{A}(P).$$

Proof. The definition of $\mathcal{S}(\Pi)$ ensures that the graph homomorphisms $f^* : G^{\mathcal{S}} \to G^{\mathcal{C}}$, $g^* : G^{\mathcal{S}} \to G^{\mathcal{B}}$ establish indeed Θ-sketch morphisms $f^* : \mathcal{S} \to \mathcal{C}$ and $g^* : \mathcal{S} \to \mathcal{B}$, respectively.

Further, there exists for all Θ-sketch morphisms $f' : \mathcal{X} \to \mathcal{C}$ and $g' : \mathcal{X} \to \mathcal{B}$ with $g'; f = f'; g$ a unique graph homomorphism $k : G^{\mathcal{X}} \to G^{\mathcal{S}}$ with $k; f^* = f'$ and $k; g^* = g'$.



For any $P \in \Pi$ and any $(P,\sigma) \in \mathcal{X}(P)$ this entails $(P,\sigma;k;f^*) = (P,\sigma;f') \in \mathcal{C}(P)$ and $(P,\sigma;k;g^*) = (P,\sigma;g') \in \mathcal{B}(P)$ since f' and g', respectively, are Θ-sketch morphisms. Moreover, we have $(P,\sigma;k;f^*;g) = (P,\sigma;k;g^*;f) \in \mathcal{A}(P)$ since g and f are Θ-sketch morphisms. According to our definition this means $(P,\sigma;k) \in \mathcal{S}(P)$, thus we have shown, finally, that k defines a Θ-sketch morphism $k : \mathcal{X} \to \mathcal{S}$. □

The empty graph and and the "singleton graph" Loop provide, moreover, also the initial and the final sketch, respectively.

Definition 9 (Empty and Singleton Sketch)**.** *By $\mathbb{O}$ we denote the **empty Θ-sketch** given by the empty graph. And by $\mathbb{T}$ we denote the **singleton Θ-sketch** with the carrier graph* `Loop` *and with $\mathbb{T}(P)$ containing only the unique graph homomorphism from $\alpha(P)$ into* `Loop` *for all predicate symbols $P \in \Pi$.*

We obtain immediately

Corollary 1 (Initial and Final Sketch)**.** *The empty Θ-sketch $\mathbb{O}$ and the singleton Θ-sketch $\mathbb{T}$ are the initial and the final Θ-sketch, respectively.*

By standard categorical arguments we are able, in such a way, to construct any finite limit or colimit of sketches.

Theorem 1 ((Co)Completeness)**.** *The category $\mathbf{Ske}(\Theta)$ is finitely complete and finitely cocomplete.*

4 Indexed Semantics of Sketches

Following the "culture of theoreticians" we can define the semantics of sketches in an indexed way. First, we have to decide for a **"semantic universe"**, i.e., we have to chose an appropriate "big" graph U. In a second step we have to convert the graph U into a "semantic Θ-sketch" $\mathcal{U} = (U, \mathcal{U}(\Pi))$, i.e., for all labels $P \in \Pi$ we have to give a mathematical exact definition of the "property P", i.e., of the set

$$\mathcal{U}(P) \subset \mathsf{GRAPH}(\alpha(P), U)$$

of all "semantic valid diagrams" of arity $\alpha(P)$ where GRAPH is the category of "big" graphs. Then we can define models within this chosen "semantic universe".

Definition 10 (Semantic Universe)**.** *A **semantic Θ-universe** is a Θ-sketch $\mathcal{U} = (U, \mathcal{U}(\Pi))$.*

Remark 5 (Model Morphisms)**.** *If we want to have also morphisms between models we need some more structure in the "semantic universe". One possibility to define morpisms between models is to adapt the concept of "natural transformation" [BW90, Fia05] and, in this case, it will be necessary that the arrows in the "semantic universe" can be composed. That is, we have to assume that the graph U is actually a category. Moreover, it appears to be often necessary, in practice, to allow only for a certain kind of arrows to relate the components of models, That is, we have to indicate an appropriate subcategory of U (see [WD07] for details).*

Remark 6 (Pre-defined Semantics)**.** *The predicate labes "commutative", "limit", and "colimit" are pre-defined also in the sense that their semantic interpretaion is fixed for any category. That is, their is an exact, "generic", mathematical definition of these predicates based on universal properties (and thus on composition also). This allows to develop a nice and rich "general model theory" for these predicates.*

For arbitrary "user-defined" predicates, however, such a "general model theory" can not be provided in advance, since the semantics of the new predicates has to be defined by the inventor and will be usually restricted to one or a restricted class of semantic universes. This, however, is exactly useful and needed in most application contexts. Note, moreover, that we are free to use any mathematical tool to define the semantics of our predicates, i.e., we are not restricted to the tools "universal property" and "composition" only (see Example 11).

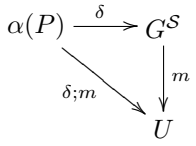
Example 6 (Semantic Universes)**.** *In case of **algebraic specifications** we can take as the semantic $\Theta_{\mathbf{AS}}$-universe the pair $\mathcal{AS} = (\mathsf{Set}, \mathcal{AS}(\Pi_{\mathbf{AS}}))$. All singleton sets will be marked with the "empty product" label $([prod], 0)$, and the labels $([prod], n)$, $n \geq 2$ are marking not only the cartesian products of n sets, but also all respective isomorphic diagrams.*

*For **equational specifications** we can extend $\mathcal{AS}$ to a semantic $\Theta_{\mathbf{EQ}}$-universe $\mathcal{EQ} = (\mathsf{Set}, \mathcal{EQ}(\Pi_{\mathbf{EQ}}))$:*

- *$\mathcal{EQ}([=])$ is the set of all diagrams δ : `Cell` $\to \mathsf{Set}$ with $\delta_1(1) = \delta_1(2)$.*
- *$\mathcal{EQ}([comp])$ is the set of all diagrams δ : `Triangle` $\to \mathsf{Set}$ with $\delta_1(3) = \delta_1(1); \delta_1(2)$.*
- *$\mathcal{EQ}([tupl], n)$ is the set of all diagrams δ : $\mathtt{Tupl}_n \to \mathsf{Set}$ with $\delta_1(\pi_i)(\delta_1(k)(e)) = \delta_1(r_i)(e)$ for all $e \in \delta_0(y)$, $i = 1, \ldots, n$.*
- *And $\mathcal{EQ}([id])$ is the set of all diagrams δ : `Loop` $\to \mathsf{Set}$ with $\delta_1(1) = id_{\delta_0(x)}$.*

The requirement, that a predicate in a specification has to become true under a semantical interpretation can be formulated now by means of the concept of sketch morphism.

Definition 11 (Models)**.** *An $\mathcal{S}$-**model** of a Θ-sketch $\mathcal{S}$ in a semantic Θ-universe $\mathcal{U}$ is a Θ-sketch morphism $m : \mathcal{S} \to \mathcal{U}$, i.e., $(P, \delta : \alpha(P) \to G^{\mathcal{S}}) \in \mathcal{S}(P)$ implies $(P, \delta; m : \alpha(P) \to U) \in \mathcal{U}(P)$ for all $P \in \Pi$.*



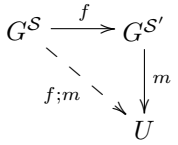
By $Mod(\mathcal{S},\mathcal{U})$ we denote the collection of all $\mathcal{S}$-models in $\mathcal{U}$.

Example 7 (Models)**.** *Models of $\Theta_{\mathbf{AS}}$-sketches and $\Theta_{\mathbf{EQ}}$-sketches correspond to Σ-algebras. We can define, for example, the "natural numbers" as a $\mathcal{S}_{\Sigma_{Nat}}$-model $nat : \mathcal{S}_{\Sigma_{Nat}} \to \mathcal{AS}$ given by $nat_0(N) = \mathbb{N}$, $nat_0(M) = \mathbb{N} \times \mathbb{N}$, $nat_1(z) = 0$, $nat_1(s) = (\_ + 1)$, $nat_1(p) = (\_ + \_)$, and two projections $nat_1(\pi_i) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $i = 1, 2$. Since the equation $(p(x, s(y)) = s(p(x, y))$ is satisfied for natural numbers this $\mathcal{S}_{\Sigma_{Nat}}$-model can be extended uniquely to a $\mathcal{S}_{SP_{Nat}}$-model.*

"Binary digits" give rise to another $\mathcal{S}_{\Sigma_{Nat}}$-model $bin : \mathcal{S}_{\Sigma_{Nat}} \to \mathcal{AS}$ given by $bin_0(N) = \mathbf{2} = \{0, 1\}$, $bin_0(M) = \mathbf{2} \times \mathbf{2}$, $bin_1(z) = 0$, $bin_1(s) = (\_ + 1 \bmod 2)$, $bin_1(p) = (\_ + \_ \bmod 2)$, and two projections $bin_1(\pi_i) : \mathbf{2} \times \mathbf{2} \to \mathbf{2}$, $i = 1, 2$. □

Since our concepts are defined in an indexed way we get different kinds of model transformations for free. Any "specification morphism", for example, gives rise, by pre-composition, to a transformation of models into the opposite direction.

Proposition 3 (Reduction Map)**.** *For any Θ-sketch morphism $f : \mathcal{S} \to \mathcal{S}'$ we obtain a **reduction map** $f_{\mathcal{U}} : Mod(\mathcal{S}',\mathcal{U}) \to Mod(\mathcal{S},\mathcal{U})$ where $f_{\mathcal{U}}(m) \stackrel{def}{=} f; m$ for any $\mathcal{S}'$-model $m : \mathcal{S}' \to \mathcal{U}$.*



5 Sketch Transformations and Sketch Operations

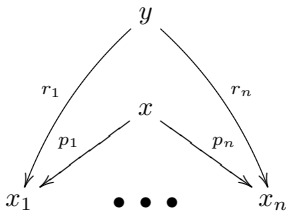
In analogy to graph transformations [EEPT06] we can consider a sketch morphism as a simple non-deleting transformation rule.

Definition 12 (Sketch Production)**.** *A Θ-**sketch production** is given by a Θ-sketch morphism $r : \mathcal{L} \to \mathcal{R}$.*

Example 8 (Productions for Term Construction)**.** *To model the inductive construction of Σ-terms over finite sets of variables we could use the following $\Theta_{\mathbf{EQ}}$-sketch productions:*

- *The production $[id\rangle : \mathcal{L}_{[id\rangle} \hookrightarrow \mathcal{R}_{[id\rangle}$ introduces identity requirements. $\mathcal{L}_{[id\rangle}$ is given by the graph* `Node` *and $\mathcal{R}_{[id\rangle}$ is given by the graph* `Loop` *and the $[id]$-marked diagram $id_{\texttt{Loop}}$.*
- *The production $[comp\rangle : \mathcal{L}_{[comp\rangle} \hookrightarrow \mathcal{R}_{[comp\rangle}$ generates the composition of two arrows. $\mathcal{L}_{[comp\rangle}$ is given by the graph $(x \stackrel{1}{\to} y \stackrel{2}{\to} z)$ and $\mathcal{R}_{[comp\rangle}$ is given by the graph* `Triangle` *and the $[comp]$-marked diagram $id_{\texttt{Triangle}}$.*
- *The productions $([prod\rangle, n) : \mathcal{L}_{([prod\rangle,n)} \hookrightarrow \mathcal{R}_{([prod\rangle,n)}$ introduce products of n nodes. $\mathcal{L}_{([prod\rangle,n)}$ is given by the discrete graph $(x_1 \dots x_n)$ and $\mathcal{R}_{([prod\rangle,n)}$ is given by the graph $\texttt{Span}_n$ and the $([prod], n)$-marked diagram $id_{\texttt{Span}_n}$.*
- *And the productions $([tupl\rangle, n) : \mathcal{L}_{([tupl\rangle,n)} \hookrightarrow \mathcal{R}_{([tupl\rangle,n)}$ describe the **tupling** of n arrows. $\mathcal{L}_{([tupl\rangle,n)}$ is*

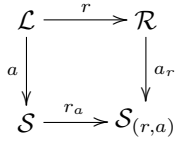
given by the following graph $\texttt{Term}_n$



and the $([prod], n)$*-marked diagram* $in_n^{Term} : \texttt{Span}_n \hookrightarrow \texttt{Term}_n$*. Finally,* $\mathcal{R}_{([tupl\rangle,n)}$ *is given by the graph* $\texttt{Tupl}_n$*, the* $([tupl], n)$*-marked diagram* $id_{\texttt{Tupl}_n}$*, the* $([prod], n)$*-marked diagram* $in_n^{Tupl} : \texttt{Span}_n \hookrightarrow \texttt{Tupl}_n$*, and* n $[comp]$*-marked diagrams* $\delta_i : \texttt{Triangle} \to \texttt{Tupl}_n$*,* $i = 1, \ldots, n$ *with* $\delta_{i,0}(x) = y$*,* $\delta_{i,0}(y) = x$*,* $\delta_{i,0}(z) = x_i$*,* $\delta_{i,1}(1) = k$*,* $\delta_{i,1}(2) = p_i$*, and* $\delta_{i,1}(3) = r_i$*.*

The transformation of sketches by means of productions can be described by single pushout (SPO) constructions.

Definition 13 (Sketch Transformations)**.** *Given a* Θ*-sketch production* $r : \mathcal{L} \to \mathcal{R}$ *and a* Θ*-sketch* $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ *with a* Θ*-sketch morphism* $a : \mathcal{L} \to \mathcal{S}$*, called the* ***match****, a* ***direct*** Θ***-sketch transformation*** $\mathcal{S} \overset{r,a}{\Longrightarrow} \mathcal{S}_{(r,a)}$ *is given by the following pushout diagram in* $\mathbf{Ske}(\Theta)$*:*



Sketch transformations provide a mechanism to produce new sketches, in a well-defined constructive way, by identifying nodes and/or arrows and/or by adding new nodes, new arrows, and/or new marked diagrams. Such a mechanism allows us to reflect the construction of "algebraic terms" or deduction rules within the Generalized Sketch framework and, for example, to define the semantics of database query languages and view mechanisms [DK97]. Derived structures are also necessary to relate specifications and to describe data and schema integration and other model management tasks [Dis05].

Example 9 (Construction of Terms)**.** *The construction of* Σ*-terms over finite sets of variables can be simulated in the GS formalism by applying inductively* $\Theta_{\mathbf{EQ}}$*-sketch transformations using the* $\Theta_{\mathbf{EQ}}$*- productions from Example 8. (The interest reader may compare the discussion here and in Example 10 with the rule-based construction of syntactic categories as presented in [CGRW95], for example.)*

We start with a $\Theta_{\mathbf{AS}}$*-sketch* $\mathcal{S}_\Sigma$ *representing the algebraic signature* Σ*, as the* $\Theta_{\mathbf{AS}}$*-sketch* $\mathcal{S}_{\Sigma_{Nat}}$ *in Example 3, for example. Applying the productions* $([prod\rangle, n)$ *with* $n \geq 2$ *we generate sequences of sort symbols, i.e., finite sets (lists) of variables. To avoid nesting of products we could introduce a further predicate symbol* $[sort]$ *into* $\Theta_{\mathbf{EQ}}$ *with arity* $\texttt{Node}$ *to indicate the basic sort symbols of the signature. In* $\mathcal{S}_{\Sigma_{Nat}}$*, for example, we would mark only the node* N *with this predicate. By marking, further, every node in* $\mathcal{L}_{([prod\rangle,n)}$ *by this predicate we will ensure that we generate only sequences of basic sort symbols.*

Applying the productions $([tupl\rangle, n)$ *with* $n \geq 2$ *we generate tuples* $(t_1, \ldots, t_n)$ *of terms. And again, to avoid nested tuples, we could mark the nodes* $x_1, \ldots, x_n$ *in* $\mathcal{L}_{([tupl\rangle,n)}$ *by* $[sort]$*.*

New terms are finally generated by applying the production $[comp\rangle$*. To ensure that we only generate "single terms"* $op(t_1, \ldots, t_n)$ *we could extend* $\Theta_{\mathbf{EQ}}$ *by a predicate symbol* $[op]$ *with arity* $\texttt{Arrow}$ *and mark the arrow* 2 *in* $\mathcal{L}_{[comp\rangle}$ *and all the arrows in* $\mathcal{S}_\Sigma$ *by this predicate.*

Remark 7 (Construction of Names)**.** *Examples 8 and 9 shed some light on a very important* ***methodological point****: For many-sorted algebraic signatures we have the following induction rule for the construction of* Σ*-terms: For all* $op : s_1 \ldots s_n \to s$ *in* OP

$$t_1 \in T(\Sigma, X)_{s_1}, \ldots, t_n \in T(\Sigma, X)_{s_n} \quad \Rightarrow \quad op(t_1, \ldots, t_n) \in T(\Sigma, X)_s.$$

In view of Generalized Sketches this means that we construct out of n *arrows* $X \to s_i$ *with "names"* t_i *and one arrow* $s_1 \ldots s_n \to s$ *with "name" op a new arrow* $X \to s$ *with the "compound name"* $op(t_1, \ldots, t_n)$*. That*

is, traditional, "syntactic" oriented approaches to formal specifications, as algebraic specifications, are mainly concerned about the construction of denotations (names) for new items out of the denotations (names) of given items. The fact that also new "arrows" are constructed is reflected, if at all, by the "typing" of the new items. In contrast, the Generalized Sketch framework is focusing on the construction of new nodes and new arrows. There is no build-in mechanism to construct names for the new items. We are convinced that a practical useful specification formalism has to combine both mechanisms. In Example 9 we could, for example, equip the $\Theta_{\mathbf{EQ}}$*-sketch productions* $([tupl\rangle, n)$ *with a mechanism that creates for each match* $a : \mathcal{L}_{([tupl\rangle,n)} \to \mathcal{S}$ *the new denotation* $\langle t_1, \ldots, t_n\rangle$ *for the arrow* $a^*(k)$ *if* t_i *is the denotation for* $a(r_i)$*,* $i = 1, \ldots, n$*.*

Such a "naming mechanism" would allow us, for example, to control the application of productions in such a way that we generate a term only once instead of generating copies of the same term again and again.

Example 10 (Equational Logic)**.** *The deduction rules for equational logic can be modeled by productions that introduce only new diagrams marked with* $[=]$ *(compare the rules in [CGRW95]).*

The reflexivity rule, for example, can be modeled by a production $[refl\rangle$ *with* $\mathcal{L}_{[refl\rangle}$ *the graph* `Arrow` *and with* $\mathcal{R}_{[refl\rangle}$ *given by the graph* `Arrow` *and the unique* $[=]$*-marked diagram from* `Cell` *into* `Arrow`*.*

That identical arrows are right-neutral w.r.t. composition can be modeled by a production $[rn\rangle$*:* $\mathcal{L}_{[rn\rangle}$ *is given by a graph* G *consisting of the arrows* $1, 2 : x \to y$*,* $3 : y \to y$*, a* $[comp]$*-marked diagram from* `Triangle` *into* G *mapping* $1 \mapsto 1$*,* $2 \mapsto 3$*,*$3 \mapsto 2$*, and an* $[id]$*-marked diagram from* `Loop` *into* G *mapping* $1 \mapsto 3$*.* $\mathcal{R}_{[rn\rangle}$ *introduces then, in addition, a new* $[=]$*-marked diagram* $incl$: `Cell` $\to G$*.*

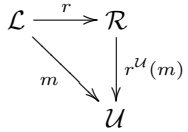
Note, that $\mathcal{S}$ *is "closed under a deduction rule"* $r : \mathcal{L} \to \mathcal{R}$ *if* $\mathcal{S} = \mathcal{S}_{(r,a)}$ *for all matches* a *of* $\mathcal{L}$ *in* $\mathcal{S}$*.*

Similar to graph transformations, also sketch transformations can be seen as pure syntactic constructions on diagrams. And, it's astonishing that the semantic interpretation of graphs and of graph productions in a semantic universe as well as the interplay between graph transformations and the semantic interpretation of productions has never been addressed in the area of Graph Transformations.

In contrast, the concept of model arises quite naturally in most applications of the GS framework. Also our running example of "equational specifications" indicates that it is natural to define an "indexed semantics" for sketch productions and sketch transformations. What we want to have, in an application, is that any model of $\mathcal{S}$ in a semantic universe $\mathcal{U}$ can be extended (sometimes in a unique way) to a model of $\mathcal{S}_{(r,a)}$ in $\mathcal{U}$. One possibility to gain this is to have a fixed interpretation of the sketch production in the semantic universe $\mathcal{U}$.

Definition 14 (Sketch Operations)**.** *For a* Θ*-sketch production* $r : \mathcal{L} \to \mathcal{R}$ *a* Θ***-sketch operation*** $r^{\mathcal{U}}$ *in a semantic* Θ*-universe* $\mathcal{U} = (U, \mathcal{U}(\Pi))$ *is a map* $r^{\mathcal{U}} : Mod(\mathcal{L}, \mathcal{U}) \to Mod(\mathcal{R}, \mathcal{U})$*.*

$r^{\mathcal{U}}$ *is called* ***strongly persistent*** *iff* $r_{\mathcal{U}}(r^{\mathcal{U}}(m)) = m$ *for any* $\mathcal{L}$*-model* $m : \mathcal{L} \to \mathcal{U}$*, i.e., iff the following diagram commutes*



Remark 8 (Persistency vs. Injectivity)**.** *For a production* $r : \mathcal{L} \to \mathcal{R}$ *and a semantic universe* $\mathcal{U}$ *where we can find for any two nodes* $x \neq y$ *(arrows* $f \neq g$*) in* $\mathcal{L}$ *a model* m *in* $\mathcal{U}$ *such that* $m(x) \neq m(y)$ $(m(f) \neq m(g))$ *the existence of at least one strongly persistent operation* $r^{\mathcal{U}}$ *implies that* r *has to be injective on nodes (arrows).*

There are, however, relevant cases where a "natural semantics" of predicates entails the non-existence of those models. The predicate $[=]$*, for example, should be naturally interpreted in any "semantic universe" as the identity of arrows (compare Example 6). In all these natural cases we can define productions* $r : \mathcal{L} \to \mathcal{R}$ *non-injective on arrows, however, with a strong persistent operation* $r^{\mathcal{U}} : Mod(\mathcal{L}, \mathcal{U}) \to Mod(\mathcal{R}, \mathcal{U})$ *as long as we ensure that arrows in* $\mathcal{L}$ *are only identified by* r *if they are marked with the predicate label* $[=]$*.*

Example 11 (Sketch Operations)**.** *We can define a sketch operation for* $[id\rangle$ *and* $[comp\rangle$ *in any semantic universe* $\mathcal{U} = (U, \mathcal{U}(\Pi))$ *where* U *is the underlying graph of a category* U*. For any* $\mathcal{L}_{[id\rangle}$*-model* $m : \mathcal{L}_{[comp\rangle} \to \mathcal{U}$*, i.e., for any object in* U *we can define* $[id\rangle^{\mathcal{U}}(m)(1) = id_{m(x)}$*. And for any* $\mathcal{L}_{[comp\rangle}$*-model* $m : \mathcal{L}_{[comp\rangle} \to \mathcal{U}$ *we can define by composition in* U $[comp\rangle^{\mathcal{U}}(m)(3) = m(1); m(2)$*.*

Similar we can define a sketch operation for $([prod\rangle, n)$ *in any semantic universe* $\mathcal{U}$ *with* U *equal to* Set*,* Par*, or* Pow *by the cartesian product of sets, i.e., for any* $\mathcal{L}_{([prod\rangle,n)}$*-model* $m : \mathcal{L}_{([prod\rangle,n)} \to \mathcal{U}$ *we can define* $([prod\rangle, n)^{\mathcal{U}}(m)(x) = m(x_1) \times \cdots \times m(x_n)$*.*

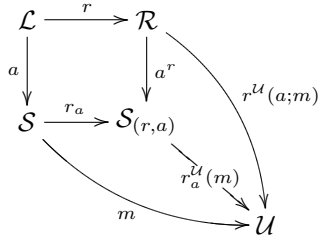
Note, that we have to make a choice between all the isomorphic possibilities to define products. Note further, that the cartesian product of sets is ***not*** *the categorical product in* Par *or* Pow*, respectively. That is, it can not be described by a universal property in these categories.*

Since the definition of $([tupl\rangle, n)$ *reflects the property of categorical products we can definine, in contrast, a sketch operation for* $([tupl\rangle, n)$ *only in* Set*. To model the tupling of partial functions, for example, we could modify* $([tupl\rangle, n)$ *replacing the* $[comp]$*-marked in* $\mathcal{R}_{([tupl\rangle,n)}$ *by diagrams expessing inequalities of partial functions (compare [Rei87, Wol90, CGRW95]).*

The universal property of pushouts ensures that sketch transformations and persistent sketch operations are compatible, i.e., any syntactic transformation induces a corresponding semantic transformation of models.

Theorem 2 (Compatibility)**.** *Given a* Θ*-sketch production* $r : \mathcal{L} \to \mathcal{R}$ *and a strongly persistent* Θ*-sketch operation* $r^{\mathcal{U}} : Mod(\mathcal{L},\mathcal{U}) \to Mod(\mathcal{R},\mathcal{U})$ *any match* $a : \mathcal{L} \to \mathcal{S}$ *into a* Θ*-sketch* $\mathcal{S}$ *induces a strongly persistent* Θ*-sketch operation* $r^{\mathcal{U}}_a : Mod(\mathcal{S},\mathcal{U}) \to Mod(\mathcal{S}_{(r,a)},\mathcal{U})$*.*

Proof. For any $\mathcal{S}$-model $m : \mathcal{S} \to \mathcal{U}$ we have a $\mathcal{R}$-model $r^{\mathcal{U}}(a; m) : \mathcal{R} \to \mathcal{U}$ such that $a; m = r; r^{\mathcal{U}}(a; m)$ due to the strong persistency of $r^{\mathcal{U}}$. Now, the pushout property provides a (unique) $\mathcal{S}_{(r,a)}$-model $r^{\mathcal{U}}_a(m) : \mathcal{S}_{(r,a)} \to \mathcal{U}$ such that $r_a; r^{\mathcal{U}}_a(m) = m$ (and $a_r; r^{\mathcal{U}}_a(m) = r^{\mathcal{U}}(a; m)$).



□

Remark 9 (Amalgamation)**.** *The construction of the mediating morphism* $r^{\mathcal{U}}_a(m)$ *corresponds to what is called the* ***amalgamated sum*** *of algebras in Algebraic Specifications [EM85]. And the statement in Theorem 2 corresponds to the first part of the* ***extension lemma*** *in [EM85]. It is straightforward to prove the full ammalgamation and extension lemmata for Generalized Sketches (even for signatures with dependencies). We don't present these results here since we concentrate on the basic mechanisms for translating indexed into fibred semantics.*

6 Grothendieck Construction

The indexed semantics defined in the last two sections works well and we can build a lot of nice theory based on it (see [WD07]). For example, we could prove the full "extension lemma", we could show that the GS formalism provides an "institution" [GB92], and so on. Trying, however, to apply this theory in practice we are faced with a severe cultural gap. Practitioners (and especially software engineers) are living in a "fibred world". Instead of considering **models** in our sense, i.e., interpretations of a syntactic structure G in a semantic universe as, for example, graph homomorphisms $m : G \to U$ they prefer to talk about and to think in terms of **instances** $\tau : I \to G$. And, even more, the structure G is considered in this context to be a "model" of I[3]. To bridge the cultural gap we have to transfer our indexed definitions, constructions and statements in to the fibred world.

Those transitions can be based on the so-called Grothendieck construction [BW90, MWH07]. The idea is to construct a "flat structure" build up from the disjoint union of all the single "semantic objects" $m(i)$ with i a node in G plus a coding of all the single "semantic translations" $m(\sigma) : m(i) \to m(j)$ for arrows $\sigma : i \to j$ in G.

A more detailed and fine-grained investigation of the different variants of this construction for different types of syntactic structures G and for different semantic universes is planned for a forthcoming paper. Since we are dealt here with generalized sketches we will consider only graphs as syntactic structures and we will restrict ourself to the category Graph as the only semantic universe.

Definition 15 (Grothendieck Construction)**.** *For any small graph* G *and any graph homomorphism* $m : G \to$ Graph *we can construct a small graph* $Fl(m)$ *as follows:*

[3]Note, that such a use of the term "model" in the sense of a simplifying abstraction corresponds to the traditional use of this term in physics, for example.

- ***nodes:*** *$\langle i, x\rangle$ with i a node in G and x a node in $m(i)$*
- ***arrows:***
  1. *$\langle i, f\rangle : \langle i, x\rangle \to \langle i, y\rangle$ with i a node in G and $f : x \to y$ an arrow in $m(i)$.*
  2. *$\langle \sigma, x\rangle : \langle i, x\rangle \to \langle j, m(\sigma)(x)\rangle$ with $\sigma : i \to j$ an arrow in G and x a node in $m(i)$ (and thus $m(\sigma)(x)$ a node in $m(j)$).*

$$i \xrightarrow{\quad\sigma\quad} j$$

$$m(i) \xrightarrow{\quad m(\sigma)\quad} m(j)$$

$$\begin{array}{ccc} x & \longmapsto & m(\sigma)(x) \\ {\scriptstyle f}\downarrow & & \\ y & & \end{array}$$

We will call the arrows $\langle i, f\rangle$ **internal arrows** since they represent arrows within the single "semantic objects" $m(i)$ and the arrows $\langle \sigma, x\rangle$ will be called **external arrows** since they represent the graphs of the "semantic translations" $m(\sigma) : m(i) \to m(j)$.

Remark 10 (Grothendieck Construction for Categories)**.** *The traditional Grothendieck construction (see [BW90, MWH07]) flattens a functor $m : G \to \mathsf{Cat}$ into a category $Fl(m)$ where the arrows in $Fl(m)$ are obtained by composing internal and external arrows in any order. The crucial point with categories, however, is that any such sequence of composed internal and external arrows can be "normalized" into the composition of a single external arrow with a single internal arrow:*

Any sequence of external arrows reduces to a single external arrow by composition in G and due to the functor property of m (i.e., we have $m(\sigma;\tau)(x) = m(\tau)(m(\sigma)(x))$ in the second diagram below). Any sequence of internal arrows normalizes to a single internal arrow by composition in the corresponding component $m(i)$. And, finally, the composition of a single internal arrow with a single external arrow can be replaced by an equivalent composition of an external arrow with an internal arrow since the "semantic translations" $m(\sigma) : m(i) \to m(j)$ are functors (see the square in the second diagram below). Based on this normalization the arrows in $Fl(m)$ can be defined, therefore, as pairs

$$\langle \sigma, f\rangle : \langle i, x\rangle \to \langle j, y\rangle$$

with $\sigma : i \to j$ an arrow in G and $f : m(\sigma)(x) \to y$ an arrow in $m(j)$.

$$i \xrightarrow{\quad\sigma\quad} j$$

$$m(i) \xrightarrow{\quad m(\sigma)\quad} m(j)$$

$$\begin{array}{ccc} x & \longmapsto & m(\sigma)(x) \\ & & {\scriptstyle f}\downarrow \\ & & y \end{array}$$

Composition in $Fl(m)$ is defined, following the described "normalization procedure", by

$$\langle \sigma, f\rangle ; \langle \tau, g\rangle = \langle \sigma;\tau, m(\tau)(f); g\rangle$$

$$i \xrightarrow{\sigma} j \xrightarrow{\tau} k$$

$$m(i) \xrightarrow{m(\sigma)} m(j) \xrightarrow{m(\tau)} m(k)$$

$$\begin{array}{ccccc} x & \longmapsto & m(\sigma)(x) & \longmapsto & m(\sigma;\tau)(x) \\ & & \Big\downarrow{\scriptstyle f} & & \Big\downarrow{\scriptstyle m(\tau)(f)} \\ & & y & \longmapsto & m(\tau)(y) \\ & & & & \Big\downarrow{\scriptstyle g} \\ & & & & z \end{array}$$

It is worth to mention, that in case of categories the "external arrows" can be axiomatically characterized by a universal property as so-called ***(op)cartesian arrows****. The uniqueness part of this property includes, moreover, an implicit logical coding of the assignments $f \mapsto m(\sigma)(f)$. In case of graphs this information is lost in the flat structure $Fl(m)$. It may be reasonable to extend $Fl(m)$ to a sketch with predicates $[int]$, $[ext]$ for arrows and a corresponding predicate $[mapsto]$ for squares to represent this lost information.*

Note, that the Grothendieck construction is presented traditionally in the literature as a construction for functors $m : G^{op} \to \mathsf{Cat}$.

Remark 11 (Opposite Graphs)**.** *There are four possible variants of the Grothendieck construction for graphs. Out of Definition 15 we obtain the three other variants by replacing G by G^{op} and/or $m(i)$ by $m(i)^{op}$ for all nodes i in G.*

Changing G to G^{op} means to inverse the external arrows in $Fl(m)$ and changing $m(i)$ to $m(i)^{op}$ inverses the internal arrows. In such a way, the resulting graph for the variant $(G, m(i))$, the one in Definition 15, will be opposite to the resulting graph for the variant $(G^{op}, m(i)^{op})$, and the result for $(G, m(i)^{op})$ will be opposite to the result for $(G^{op}, m(i))$. This means that we have "up to opposition" two variants $(G, m(i))$ and $(G, m(i)^{op})$, and they produce, in general, non-isomorphic results. We have, therefore, to decide which variant is the appropriate one in the corresponding application context.

Projecting out the first component of the nodes and arrows should provide now a graph homomorphism from $Fl(m)$ into G. This means, however, that we have to map the internal arrows $\langle i, f\rangle$ to an arrow from i to i in G. To do so, we have to introduce new designated identical loops in G.

Definition 16 (Identical Arrows)**.** *Given a graph G we define a new graph G^{id} with*

- $G_0^{id} = G_0$,
- $G_1^{id} = G_1 \cup \{id_i \mid i \in G_0\}$,
- $sc^{G^{id}} \downarrow G_1 = sc^G$,
- $tr^{G^{id}} \downarrow G_1 = tr^G$, *and*
- $sc^{G^{id}}(id_i) = tr^{G^{id}}(id_i) = i$ *for all* $i \in G_0$.

Any graph homomorphism $\varphi : G \to H$ can be extended to a graph homomorphism $\varphi^{id} : G^{id} \to H^{id}$ with

- $\varphi_0^{id} = \varphi_0$,
- $\varphi_1^{id} \downarrow G_1 = \varphi_1$, *and*
- $\varphi_0^{id}(id_i) = id_{\varphi_0(i)}$ *for all* $i \in G_0$.

Now we can define the intended projection.

Definition 17 (Projection)**.** *For any graph G and any graph homomorphism $m : G \to \mathsf{Graph}$ we obtain a graph homomorphism $pr_m : Fl(m) \to G^{id}$ as follows:*

- $pr_m(\langle i, x\rangle) = i$ *for any node* $\langle i, x\rangle$ *in* $Fl(m)$,
- $pr_m(\langle i, f\rangle) = id_i$ *for any arrow* $\langle i, f\rangle$ *in* $Fl(m)$,
- $pr_m(\langle \sigma, x\rangle) = \sigma$ *for any arrow* $\langle \sigma, x\rangle$ *in* $Fl(m)$.

Remark 12 (Opfibrations)**.** *Note, that the projections are "opfibrations", i.e., they are "backwards reflecting" in the sense, that for any arrow* $\sigma : i \to j$ *in* G *and any node* $\langle i, x\rangle$ *in* $Fl(m)$ *there is a unique "external arrow"* $\langle \sigma, x\rangle$ *such that* $pr_m(\langle \sigma, x\rangle) = \sigma$*. The existence of such an arrow reflects the totality of the "semantical translations" and the uniqueness reflects that these translations are indeed functions.*

It is folklore that the Grothendieck construction turns composition, i.e., commutative triangles, into pullback diagrams. Or, to make it more precise: Considering the category GRAPH of "big graphs" and the full embedding functor em : Graph $\to$ GRAPH we can define the comma category $(em \downarrow \mathsf{Graph})$. Objects (G, m) in $(em \downarrow \mathsf{Graph})$ are the graph homomorphisms $m : G \to \mathsf{Graph}$ from Definition 15 and morphism $\varphi : (H, n) \to (G, m)$ are given by graph homomorphisms $\varphi : H \to G$ such that $\varphi; m = n$, i.e., by commutative triangles in GRAPH.

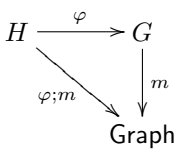
Let's further consider the arrow category $\mathsf{Graph}^{\to}$. Objects (A, h, B) are given by graph homomorphisms $h : A \to B$ and morphisms $(a, b) : (A_1, h_1, B_1) \to (A_2, h_2, B_2)$ are commutative squares in Graph, i.e., graph homomorphisms $a : A_1 \to A_2$ and $b : B_1 \to B_2$ such that $h_1; b = a; h_2$.

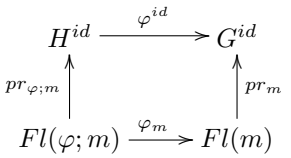
Definition 17 states then that the Grothendieck construction maps objects in $(em \downarrow \mathsf{Graph})$ to objects in $\mathsf{Graph}^{\to}$. And this mapping can be extended to arrows.

Proposition 4 (Commutative Triangles to Pullbacks)**.** *For any homomorphism* $\varphi : H \to G$ *between small graphs and any graph homomorphism* $m : G \to$ Graph *the assignments*

- $\varphi_m(\langle i, x\rangle) = \langle \varphi(i), x\rangle$ *for any node* $\langle i, x\rangle$ *in* $Fl(\varphi, m)$,
- $\varphi_m(\langle i, f\rangle) = \langle \varphi(i), f\rangle$ *for any arrow* $\langle i, f\rangle$ *in* $Fl(\varphi, m)$,
- $\varphi_m(\langle \sigma, x\rangle) = \langle \varphi(\sigma), x\rangle$ *for any arrow* $\langle \sigma, x\rangle$ *in* $Fl(\varphi, m)$.

define a graph homomorphism $\varphi_m : Fl(\varphi, m) \to Fl(m)$ *such that the following right diagram is a pullback diagram in* Graph



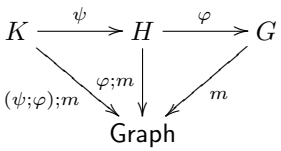


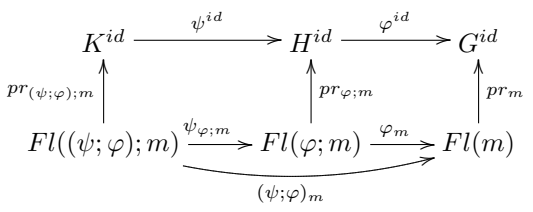
Proof. Pullbacks in Graph are constructed componentwise for nodes and edges in Set and it is easy to verify that our definition just mimics the standard construction of pullbacks in Set by constructing a product and then an equalizer. □

Composition in $(em \downarrow \mathsf{Graph})$ is defined by composition in Graph and is well-defined since composition in Graph is associative and since em is a functor.

Corollary 2 (Composition of Pullbacks)**.** *For any homomorphisms* $\psi : K \to H$, $\varphi : H \to G$ *between small graphs and any graph homomorphism* $m : G \to$ Graph *we have*

$$Fl((\psi;\varphi); m) = Fl(\psi; (\varphi; m)), \quad pr_{(\psi;\varphi);m} = pr_{\psi;(\varphi;m)} \quad \textit{and} \quad (\psi;\varphi)_m = \psi_{\varphi;m}; \varphi_m$$



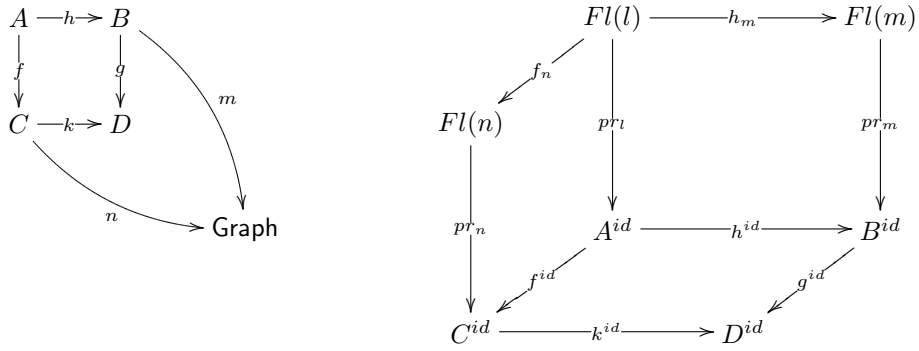


Proof. Due to the associativity of composition in GRAPH, i.e., due to $(\psi; \varphi); m = \psi; (\varphi; m)$, this follows immediately from Definition 17 and the fact $(\psi; \varphi)^{id} = \psi^{id}; \varphi^{id}$. □

Theorem 3 (Grothendieck Functor)**.** *The assignments* $(G, m) \mapsto pr_m$ *and* $\varphi \mapsto (\varphi_m, \varphi^{id})$ *according to Definition 15 and Proposition 4, respectively, define a functor* $Fl : (em \downarrow \mathsf{Graph}) \to \mathsf{Graph}^{\to}$

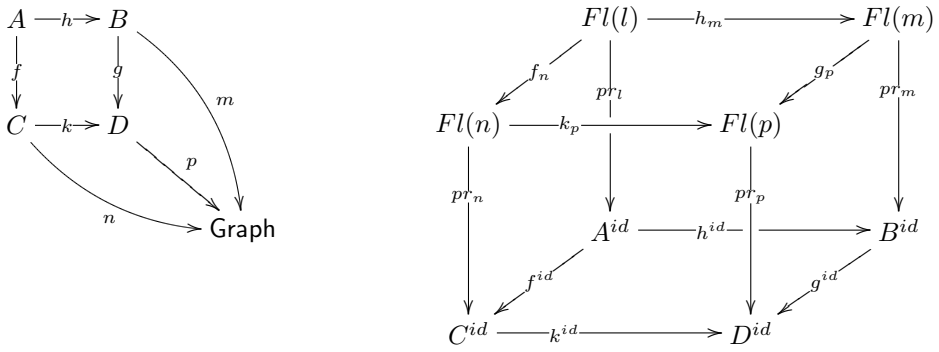
Proof. Follows immediately from Corollary 2 and the facts $(id_G)^{id} = id_{G^{id}}$ and $(id_G)_m = id_{Fl(m)}$. □

Now we want to analyze the argumentation in Theorem 2 in view of the Grothendieck construction. That is, we consider a pushout diagram in Graph and two graph homomorphisms $m : B \to \mathsf{Graph}$ and $n : C \to \mathsf{Graph}$ such that $h; m = f; n (=: l)$:



Applying the Grothendieck construction, we obtain a half cube with a pushout in the bottom and with pullbacks in the left face and the back face.

The **existence** of a $p : D \to \mathsf{Graph}$ with $g; p = m$ and $k; p = n$ means now that there is a completion of the half cube to a commutative cube with the front face and the rightface pullbacks, where the commutativity of the top face is ensured by Corollary 2.



How is the **uniqueness** of $p : D \to \mathsf{Graph}$ reflected on the fibred side? The bottom face of the cube is a pushout. Since any pushout in Graph is a semi VK square, also the top square becomes a pushout. And, since pushouts are determined "up to isomorphism", this means that there is "up to isomorphism" only one **"pullback completion"** of the half cube.

What about the other implication for VK squares? Or, to formulate it in another way: Why should pullbacks be stable under combined pushouts and pullbacks? In view of our context here this seems to be not necessary. There is, however, a subtle technological (or psychological?) point. The construction of the unique mediating morphism $p : D \to \mathsf{Graph}$ is common and looks "natural". In contrast, the construction of a "unique pullback completion" looks strange and "unnatural" even if it is the fibred counterpart of the construction of a mediating morphism.

The only "natural way" to complete the half cube seems to be to construct, first, the pushout on the top face thus, in a second step, the missing vertical arrow can be constructed as a mediating morphism for this new pushout. And, then we hope that the resulting commutative front face and right face are pullbacks as intended.

In our case, where the pushout-pullback half cube results from the Grothendieck construction both ways provide isomorphic results thus we could summarize our discussion by the slogan:

> **The Grothendieck construction transforms the statement "a square is a pushout in** Graph**" into the statement "a(nother) square is a "weak" VK square in** Graph**".**

"Weak" means that the equivalence in Definition 3 is not required to hold for all pushout-pullback half cube, but only for those one resulting from the Grothendieck construction (see the discussion of "modularity" in the concluding section).

In the context of sketch operations and graph transformations it seems to be reasonable to restrict ourself to injective graph homomorphisms $h : A \to B$ (compare the discussion in Remark 8). In Graph pushouts along monomorphisms are VK squares thus the "natural way" to complete the half cube works fine for these restricted cases.

A second look at our diagrams above makes apparent that we have actually constructed commutative squares in the categories $(em \downarrow \mathsf{Graph})$ and $\mathsf{Graph}^{\to}$, respectively.

$$\begin{array}{ccc} (A,l) & \xrightarrow{h} & (B,m) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ (C,n) & \xrightarrow{k} & (D,p) \end{array} \qquad \begin{array}{ccc} (Fl(l),pr_l,A^{id}) & \xrightarrow{(h_m,h^{id})} & (Fl(m),pr_m,B^{id}) \\ {\scriptstyle (f_n,f^{id})}\downarrow & & \downarrow{\scriptstyle (g_p,g^{id})} \\ (Fl(n),pr_n,C^{id}) & \xrightarrow{(k_p,k^{id})} & (Fl(p),pr_p,D^{id}) \end{array}$$

It is straightforward to show that these diagrams are pushout diagrams thus our slogan can be reformulated as:

Theorem 4 (Pushout Preservation)**.** *The functor* $Fl : (em \downarrow \mathsf{Graph}) \to \mathsf{Graph}^{\to}$ *preserves pushouts.*

7 From Indexed to Fibred Semantics

In this section we investigate how the indexed semantics of sketches can be reformulated in a fibred setting along the line provided by the Grothendieck construction(s). Since the Grothendieck construction turns composition into pullbacks this reformulation will be based on pullbacks. Therefore we introduce some notational conventions: For every arrow cospan $(x \xrightarrow{f} y \xleftarrow{g} z)$ in Graph, we choose a fixed pullback span $(z \xleftarrow{f^*} p \xrightarrow{g^*} x) = PB(f,g)$ with the following notational agreement. If f^*, g^* are "parallel" to f, g respectively, we write $f^* = PB_g(f)$ and $g^* = PB_f(g)$. Then the familiar lemma that composition of pullbacks is again a pullback takes the following form: $PB_{f_1;f_2}(g) \overset{i}{\cong} PB_{f_1}[PB_{f_2}(g)]$ with $i = i(f_1,f_2,g)$ a canonic isomorphism satisfying the corresponding coherence conditions.

First, we have to fix a semantic interpretation for the predicate symbols. In the indexed setting this was done by chosing a semantic universe $\mathcal{U} = (U, \mathcal{U}(\Pi))$, i.e., for each $P \in \Pi$ we had to define a set of "semantical valid" diagrams $\delta : \alpha(P) \to U$. For the semantic universe Graph the Grothendieck construction transforms those diagrams into arrows $pr_\delta : Fl(\delta) \to \alpha(P)^{id}$. This leads us to a fibred semantics of signatures as defined in [DW07], where we have to take into account that pullbacks are only determined "up to isomorphism".

Definition 18 (Fibred Semantics of Signatures)**.** *Given a signature* $\Theta = (\Pi, \alpha)$*, its* semantic interpretation *is a mapping* $[\![..]\!]$*, which assigns to each predicate symbol* P *a set* $[\![P]\!] \subset \{\tau \in \mathsf{Graph} \mid cod(\tau) = \alpha(P)\}$ *of* valid *instances, where* $[\![P]\!]$ *is assumed to be closed under isomorphisms:* $\tau \in [\![P]\!]$ *implies* $i;\tau \in [\![P]\!]$ *for any isomorphism* $i : O' \to O$ *in* Graph*.*

Remark 13 (Identical Loops)**.** *The considerations in Section 6 indicate that it may be necessary to work in the fibred setting with enhanced signatures where the arity graphs are equipped with additional identical loops. We ignore those enhancements here. It is, however, worth to point out that we should expect that (sketch based) models and meta-models used in a fibred setting, as in OO modeling for example, will come along with many identical loops.*

Remark 14 (Other Grothendieck Constructions)**.** *The Grothendieck construction presented in Section 6 can be easily adopted to other semantic universes as* Set*,* Par*,* Pow*, and* Cat*. The sets* $[\![P]\!]$ *could be but don't need to be restricted to arrows obtained by one (or all) of these variants of the Grothendieck construction.*

The concept of a "model of a sketch" turns into the concept of an "instance of a sketch" where the compatibility condition in Definition 11, based on composition, turns into a corresponding pullback requirement [DW07].

Definition 19 (Instances of Sketches)**.** *Let $\mathcal{S} = (G^{\mathcal{S}}, \mathcal{S}(\Pi))$ be a Θ-sketch. Its* instance *is an instance $\tau : O \to G^{\mathcal{S}}$ of the structural base, such that $\tau^* = PB_\delta(\tau) \in [\![P]\!]$ for all diagrams $\delta : \alpha(P) \to G^{\mathcal{S}}$ in $\mathcal{S}$.*

$$\begin{array}{ccc} O^* & \xrightarrow{\delta^*} & O \\ {\scriptstyle \tau^*}\downarrow & & \downarrow{\scriptstyle \tau} \\ \alpha(P) & \xrightarrow{\delta} & G^{\mathcal{S}} \end{array}$$

It gives us the set of all instances $Inst(\mathcal{S})$ of sketch $\mathcal{S}$.

In other words, an instance of some model satisfies a labeled diagram/formula over this model if the part of this instance over the diagram is a valid instance of its label.

Reduction maps for sketch morphisms are defined in the indexed setting by simple pre-composition. In the fibred setting we have to use the more involved pullback construction for the same purpose.

Proposition 5 (Pullbacking of Instances)**.** *If $f : \mathcal{S} \to \mathcal{S}'$ is a Θ-sketch morphism and $\tau : O \to G^{\mathcal{S}'}$ is an instance of $\mathcal{S}'$, then $\tau^* : O^* \to G^{\mathcal{S}}$ with $\tau^* = PB_f(\tau)$ is an instance of $\mathcal{S}$.*

$$\begin{array}{ccccc} O^{**} & \xrightarrow{\delta^*} & O^* & \xrightarrow{f^*} & O \\ {\scriptstyle \tau^{**}}\downarrow & & \downarrow{\scriptstyle \tau^*} & & \downarrow{\scriptstyle \tau} \\ \alpha(P) & \xrightarrow{\delta} & G^{\mathcal{S}} & \xrightarrow{f} & G^{\mathcal{S}'} \end{array}$$

Proof. For any diagram $\delta : \alpha(P) \to G^{\mathcal{S}}$ in $\mathcal{S}$ $\delta; f : \alpha(P) \to G^{\mathcal{S}'}$ becomes a diagram in $\mathcal{S}'$ since $f : \mathcal{S} \to \mathcal{S}'$ is a Θ-sketch morphism. Since τ is an instance of $\mathcal{S}'$ we know that $PB_{\delta;f}(\tau) \in [\![P]\!]$. The composition of two pullbacks is again a pullback thus $PB_{\delta;f}(\tau)$ and $\tau^{**} = PB_\delta(\tau^*)$ are isomorphic. This means that $\tau^{**} \in [\![P]\!]$ since $[\![P]\!]$ is closed under isomorphisms. □

In such a way we obtain in the fibred setting a "pullback map".

Proposition 6 (Pullback Map)**.** *For a given semantic interpretation $[\![..]\!]$ of Θ any Θ-sketch morphism $f : \mathcal{S} \to \mathcal{S}'$ defines a **pullback map** $f_{[\![..]\!]} : Inst(\mathcal{S}') \to Inst(\mathcal{S})$ with $f_{[\![..]\!]}(\tau) = \tau^* = PB_f(\tau)$ for any instance $\tau : O \to G^{\mathcal{S}'}$ of $\mathcal{S}'$.*

Based on Proposition 6 the fibred version of sketch operations can be defined striaghtforwardly.

Definition 20 (Fibred Sketch Operation)**.** *For a Θ-sketch production $r : \mathcal{L} \to \mathcal{R}$ a **fibred Θ-sketch operation** for a given semantic interpretation $[\![..]\!]$ of Θ is a map $r^{[\![..]\!]} : Inst(\mathcal{L}) \to Inst(\mathcal{R})$.*

*$r^{[\![..]\!]}$ is called **strongly persistent** iff $r_{[\![..]\!]}(r^{[\![..]\!]}(\tau)) = \tau$ for any instance $\tau : O \to G^{\mathcal{L}}$ of $\mathcal{L}$.*

*$r^{[\![..]\!]}$ is called **persistent** iff $r_{[\![..]\!]}(r^{[\![..]\!]}(\tau)) \overset{i}{\cong} \tau$ for any instance $\tau : O \to G^{\mathcal{L}}$ of $\mathcal{L}$.*

The question we want to address now is if (and how) the compatibility of sketch transformations and sketch operations can be also obtained in the fibred setting.

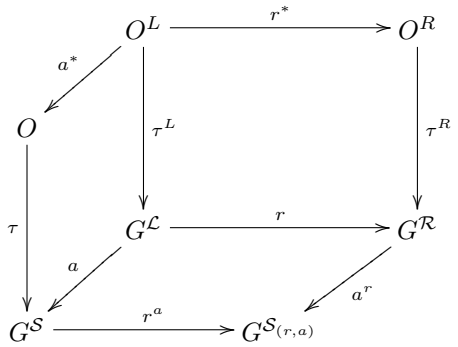
Let us consider a Θ-sketch production $r : \mathcal{L} \to \mathcal{R}$ and a direct Θ-sketch transformation $\mathcal{S} \overset{r,a}{\Longrightarrow} \mathcal{S}_{(r,a)}$, i.e., the following pushouts in $\mathbf{Ske}(\Theta)$ and Graph, respectively:

$$\begin{array}{ccc} \mathcal{L} & \xrightarrow{r} & \mathcal{R} \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a_r} \\ \mathcal{S} & \xrightarrow{r_a} & \mathcal{S}_{(r,a)} \end{array} \qquad \begin{array}{ccc} G^{\mathcal{L}} & \xrightarrow{r} & G^{\mathcal{R}} \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle a_r} \\ G^{\mathcal{S}} & \xrightarrow{r_a} & G^{\mathcal{S}_{(r,a)}} \end{array}$$

The question is if any (strongly) persistent fibred Θ-sketch operation $r^{[\![..]\!]} : Inst(\mathcal{L}) \to Inst(\mathcal{R})$ induces a (strongly) persistent fibred Θ-sketch operation $r_a^{[\![..]\!]} : Inst(\mathcal{S}) \to Inst(\mathcal{S}_{(r,a)})$ or not:

For any instance $\tau : O \to G^{\mathcal{S}}$ of $\mathcal{S}$ we get, by applying the pullback map $a_{[\![..]\!]}$, an instance $\tau^L : O^L \to G^{\mathcal{L}}$ of $\mathcal{L}$ and by applying a strongly persistent $r^{[\![..]\!]}$ we get further an instance $\tau^R : O^R \to G^{\mathcal{R}}$ of $\mathcal{R}$ such that

$\tau^L = PB_r(\tau^R)$. What we obtain is a half cube with a pushout in the bottom and with pullbacks in the left face and the back face, thus our discussion from the end of Section 6 applies.

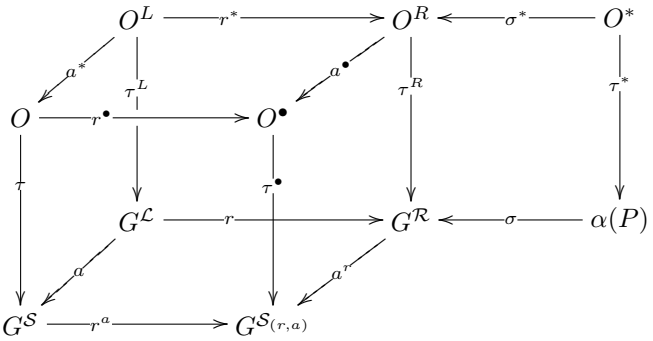


In case the half cube lifes in the image of the Grothendieck construction described in Section 6 (or of another variant) we could construct the corresponding mediating morphism on the indexed side and translate the situation back to a complete pushout-pullback cube.

In general, however, it is an **open question** if "unique pullback completions" exist for any such half cube in Graph thus we have to restrict our fibred version of compatibility to injective productions.

Theorem 5 (Fibred Compatibility)**.** *Given an injective Θ-sketch production $r : \mathcal{L} \to \mathcal{R}$ and a (strongly) persistent fibred Θ-sketch operation $r^{[\![..]\!]} : Inst(\mathcal{L}) \to Inst(\mathcal{R})$ any match $a : \mathcal{L} \to \mathcal{S}$ into a Θ-sketch $\mathcal{S}$ induces a persistent fibred Θ-sketch operation $r_a^{[\![..]\!]} : Inst(\mathcal{S}) \to Inst(\mathcal{S}_{(r,a)})$.*

Proof. We prove the statement for a strongly persistent $r^{[\![..]\!]}$. For persistent $r^{[\![..]\!]}$ the proof can be adopted easily. Consider the following cube in Graph:



The bottom face is a pushout. For any instance $\tau : O \to G^{\mathcal{S}}$ of $\mathcal{S}$ we get according to Proposition 5 an instance $\tau^L : O^L \to G^{\mathcal{L}}$ of $\mathcal{L}$ with $\tau^L = PB_a(\tau)$. Applying the strongly persistent $r^{[\![..]\!]}$ we get further an instance $\tau^R : O^R \to G^{\mathcal{R}}$ of $\mathcal{R}$ with $\tau^L = PB_r(\tau^R)$.

Constructing the pushout of the span (a^*, r^*) we obtain the two graph homomorphisms $r^\bullet : O \to O^\bullet$ and $a^\bullet : O^R \to O^\bullet$. Since $a^*;\tau;r_a = \tau^L;a;r_a = \tau^L;r;a^r = r^*;\tau^R;a^r$ there exists a unique $\tau^\bullet : O^\bullet \to G^{\mathcal{S}_{(r,a)}}$ such that $r^\bullet;\tau^\bullet = \tau;r_a$ and $a^\bullet;\tau^\bullet = \tau^R;a^r$.

Since in Graph pushouts along monomorphisms are VK squares we get, moreover, that the commutative front face and right face are pullbacks, i.e., we have $\tau \overset{i}{\cong} PB_{r_a}(\tau^\bullet)$ and $\tau^R \overset{i}{\cong} PB_{a^r}(\tau^\bullet)$. This ensures that the assignments $\tau \mapsto \tau^\bullet$ define a persistent fibred Θ-sketch operation $r_a^{[\![..]\!]} : Inst(\mathcal{S}) \to Inst(\mathcal{S}_{(r,a)})$ as long as we can show that $\tau^\bullet : O^\bullet \to G^{\mathcal{S}_{(r,a)}}$ is indeed an instance of $\mathcal{S}_{(r,a)}$: According to the construction of pushouts in Θ-sketches any diagram in $\mathcal{S}_{(r,a)}$ is of the form $\delta; r^a : \alpha(P) \to G^{\mathcal{S}_{(r,a)}}$ with $\delta : \alpha(P) \to G^{\mathcal{S}}$ a diagram in $\mathcal{S}$ or of the form $\sigma; a^r : \alpha(P) \to G^{\mathcal{S}_{(r,a)}}$ with $\sigma : \alpha(P) \to G^{\mathcal{R}}$ a diagram in $\mathcal{R}$. We consider the second case where the first case can be shown analogously. Since τ^R is an instance of $\mathcal{R}$ we know that $\tau^* = PB_\sigma(\tau^R) \in [\![\, P \,]\!]$. We have $\tau^R \overset{i}{\cong} PB_{a^r}(\tau^\bullet)$ and since pullbacks are closed under isomorphisms and since the composition of two pullbacks is again a pullback we obtain also $\tau^* \overset{i}{\cong} PB_{\sigma;a^r}(\tau^\bullet)$. This means $PB_{\sigma;a^r}(\tau^\bullet) \in [\![\, P \,]\!]$, as required, since $[\![\, P \,]\!]$ is closed under isomorphisms. □

8 Concluding Discussion and Open Questions

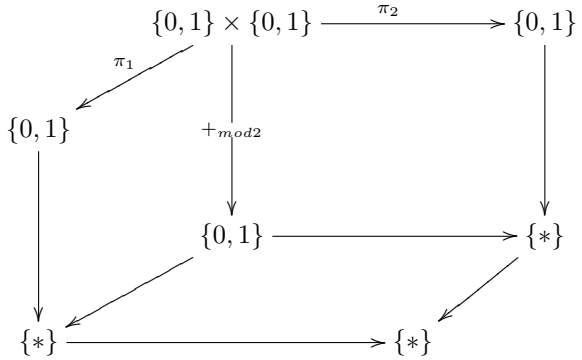
We have presented in the paper some basic definitions and results of the GS framework. Thereby it turned out that especially the syntactic part of the GS framework appears as a natural extension of the Graph Transformation framework. As a new result we have shown that the categories of sketches are finitely complete and finitely cocomplete.

Following the "indexed culture" we have given an indexed semantics for sketches and for sketch operations. Finally, we investigated how these indexed semantics can be transformed into a fibred semantics by means of (a variant of) the Grothendieck construction. Analyzing these transformations we gained some new insights into the "nature" of indexed and fibred semantics that we want to discuss in this concluding section. Especially, we are interested to shed more light on the "nature" and the rôle of the so-called van Kampen square.

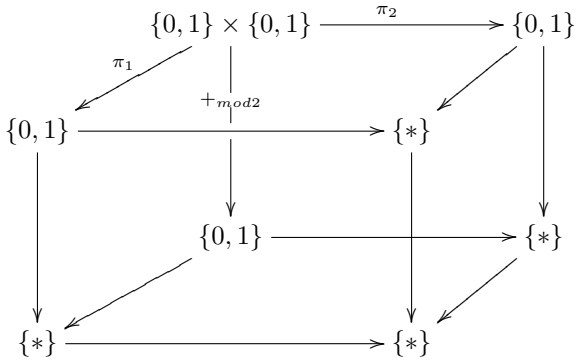
Indexed semantics turns out to be inherently modular. First, we define the syntax and then we define semantical interpretations of the syntax along the structure of the syntax. This makes indexed semantics nicely behaving and thus asking for a lot of nice theoretical results. Especially, indexed semantics makes that transformations of models and constructions of models are fully controlled/induced by syntactic transformations and constructions, respectively. We could show, for example, that the assignments $\mathcal{S} \mapsto Mod(\mathcal{S},\mathcal{U})$ and $f \mapsto f_{\mathcal{U}}$ define a functor from $\mathbf{Ske}(\Theta)$ into Cat, and we could present an institution [GB92] on top of this observation.

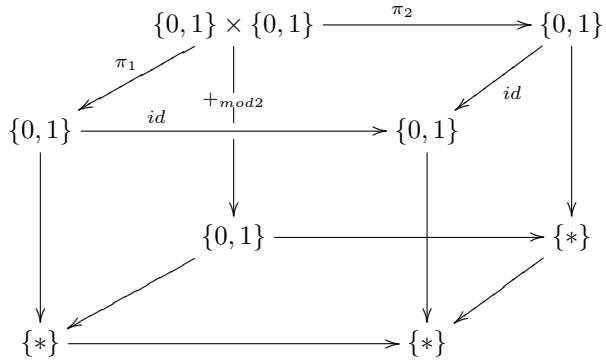
Fibred semantics, in contrast, is not inherently modular since it provides more freedom. The assignments $\mathcal{S} \mapsto Inst(\mathcal{S})$ and $f \mapsto f_{[\![..]\!]}$, for example, will provide us only with a pseudo functor from $\mathbf{Ske}(\Theta)$ into Cat since pullbacks are only determined "up to isomorphisms" (compare [DW07]).

The difference becomes more evident if we consider the "amalgamation of models". On the indexed side amalgamation is a simple and modular construction. First, we construct a pushout on the syntactic level, and then we amalgamate models by constructing a mediating morphism for this syntactic pushout. On the fibred side amalgamation of models is reflected by trying to construct a "unique pullback completion" of a pushout-pullback half cube. Such a completion, however, may not exist in many cases. So, what happens here? Let's have a look at the example for a non VK square in Set as presented in [EEPT06], p. 84. There we have the following pushout-pullback half cube:

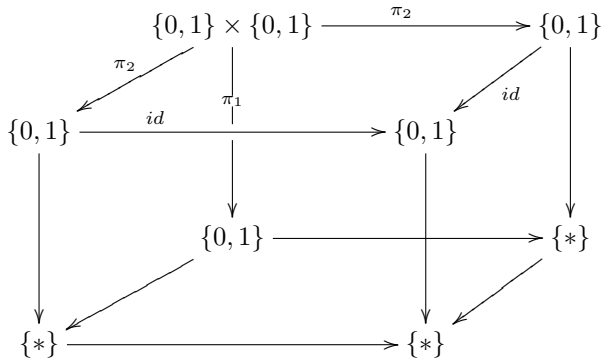


There are two chances to complete this half cube: We can construct the pushout on the top face and then we get due to the pushout property the missing unique vertical arrow making the front face and the right face commutative as shown by the left cube below. The front face and the right face, however, don't become pullbacks. On the other side we can complete the front face and the right face to pullbacks as shown by the right cube below. But, then the top face is not commutative much less a pushout.





The crucial point is that the given half cube violates the modularity between syntax and semantics. We have two different pullbacks for the same diagram, i.e., the transformation of semantics is **not** fully determined by the transformation of syntax. For indexed semantics such a situation is excluded by definition. In other words, the half cube is not in the image of the Grothendieck construction. What we get by the Grothendieck construction is the half cube in the diagram below with identical left face and back face and thus with a unique pullback completion.



We have to leave open, for a next round of discussion, the question if there is something wrong with the fibred semantics or if it is more probably wrong on the indexed side. Maybe, we have to adapt one day the interpretation: If there is a problem indeed, then the fibred semantics honestly reveals it while the indexed one hides it. Besides this, we want to formulate **three open questions** here:

1. Is there a suitable characterization of "modular" pushout-pullback half cubes, i.e., of half cubes allowing for a unique pullback completion?

2. Satisfy the "modular" pushout-pullback half cubes certain kinds of compositionality?

3. Is it reasonable to weaken the definition of VK squares by quantifying universally not over all pushout-pullbacks half cubes but only over the "modular" one?

Besides investigating these open questions it seems to be worth to provide the designers of frameworks based on categorical concepts, as the Graph Transformation and GS frameworks, and the "users" of these frameworks with a detailed and fine-grained analysis of the different variants of the Grothendieck construction for different types of syntactic structures and for different semantic universes.

The variant of the Grothendieck construction considered in this paper forced us to consider besides the structures set, graph, and category another kind of structure, namely "graphs with identical arrows". It will be interesting to see it there appear other structures on the agenda when we analyse the other variants, and it will be worth to ask if these structures are of any relevance independent of the Grothendieck construction.

References

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. Series in Computer Science. Prentice Hall International, London, 1990. 1, 3, 5, 6, 10

[CGRW95] I. Claßen, M. Große-Rhode, and U. Wolter. Categorical concepts for parameterized partial specifications. *Math. Struct. in Comp. Science*, 5(2):153–188, 1995. 9, 10, 11

[Dis96] Z. Diskin. Databases as diagram algebras: Specifying queries and views via the graph-based logic of skethes. Technical Report 9602, Frame Inform Systems, Riga, Latvia, 1996. http://citeseer.ist.psu.edu/116057.html. 1

[Dis97] Z. Diskin. Towards algebraic graph-based model theory for computer science. *Bulletin of Symbolic Logic*, 3:144–145, 1997. Presented (by title) at *Logic Colloquium'95*. 1

[Dis03] Z. Diskin. Mathematics of UML: Making the Odysseys of UML less dramatic. In H. Kilov and K. Baclawski, editors, *Practical Foundations of Business System Specifications*, pages 348–381. Kluwer Academic Publishers, 2003. 1

[Dis05] Z. Diskin. Mathematics of generic specifications for model management. In Rivero, Doorn, and Ferraggine, editors, *Encyclopedia of Database Technologies and Applications*, pages 351–366. Idea Group, 2005. 1, 5

[DK97] Z. Diskin and B. Kadish. A graphical yet formalized framework for specifying view systems. In *Advances in Databases and Information Systems*, pages 123–132, 1997. ACM SIGMOD Digital Anthology: vol.2(5), ADBIS'97. 1, 5

[DKPJ00] Z. Diskin, B. Kadish, F. Piessens, and M. Johnson. Universal Arrow Foundations for Visual Modeling. In M. Anderson, P. Cheng, and V. Haarslev, editors, *Theory and Application of Diagrams*, pages 345–360. First International Conference, Diagrams 2000, Edinburgh, Scotland, UK, September 2000, Springer, LNAI 1889, 2000. 1

[DW07] Z. Diskin and U. Wolter. Generalized Sketches: A Universal Logic for Diagrammatic Modeling in Software Engineering. *ENTCS*, 2007. Submitted. 1, 4, 7, 7, 8

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformations*. EATCS Monographs on Theoretical Computer Science. Springer, Berlin, 2006. 1, 2, 5, 8

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1985. 1, 5, 9

[Fia05] J. L. Fiadeiro. *Categories for Software Engineering*. Springer, Berlin, 2005. 2, 5

[GB92] J. A. Goguen and R. M. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journals of the ACM*, 39(1):95–146, January 1992. 6, 8

[JRW02] M. Johnson, R. Rosebrugh, and R. Wood. Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories*, 10(3):94–112, 2002. 3

[LS04] S. Lack and P. Sobociński. Adhesive Categories. In I. Walukiewicz, editor, *proceedings of FOSSACS 2004*, pages 273–288. Springer, LNCS 2987, 2004. 1

[Mak97] M. Makkai. Generalized sketches as a framework for completeness theorems. *Journal of Pure and Applied Algebra*, 115:49–79, 179–212, 214–274, 1997. 1, 4

[MWH07] A. Martini, U. Wolter, and E. H. Haeusler. Fibred and Indexed Categories for Abstract Model Theory. *Logic Journal of the IGPL*, 2007. Accepted. 1, 6, 10

[Rei87] H. Reichel. *Initial Computability, Algebraic Specifications, and Partial Algebras*. Oxford University Press, 1987. 5, 11

[WD07] U. Wolter and Z. Diskin. The Next Hundred Diagrammatic Specification Techniques – An Introduction to Generalized Sketches. Technical Report Report No 358, Department of Informatics, University of Bergen, July 2007. 1, 4, 5, 6

[Wel] C. Wells. Sketches: Outline with references. Available under http://www.cwru.edu/artsci/math/wells/pub/papers.html. 3

[Wol90] U. Wolter. An Algebraic Approach to Deduction in Equational Partial Horn Theories. *J. Inf. Process. Cybern. EIK*, 27(2):85–128, 1990. 5, 11