ALGEBRAIC DATABASES

PATRICK SCHULTZ, DAVID I. SPIVAK, CHRISTINA VASILAKOPOULOU, AND RYAN WISNESKY

ABSTRACT. Databases have been studied category-theoretically for decades. While mathematically elegant, previous categorical models have typically struggled with representing concrete data such as integers or strings.

In the present work, we propose an extension of the earlier set-valued functor model, making use of multi-sorted algebraic theories (a.k.a. Lawvere theories) to incorporate concrete data in a principled way. This approach easily handles missing information (null values), and also allows constraints and queries to make use of operations on data, such as multiplication or comparison of numbers, helping to bridge the gap between traditional databases and programming languages.

We also show how all of the components of our model—including schemas, instances, change-of-schema functors, and queries—fit into a single double categorical structure called a proarrow equipment (a.k.a. framed bicategory).

Contents

1. Introduction

Category-theoretic models of databases have been present for some time. For example in [RW92; FGR03; JR02] databases schemas are formalized as sketches of various sorts (e.g. EA sketches = finite limits + coproducts). The data itself (called an *instance*) is represented by a model of the sketch. In this language, queries can be understood as limit cones in such a sketch. While different from the traditional relational foundations of database theory [AHV95], this is in general a very natural and appealing idea.

In [Spi12], Spivak puts emphasis on the ability to move data from one format, or database schema, to another. To enable that, he proposes defining schemas to be mere categories — or in other words trivial sketches (with no (co)limit cones). A schema morphism is just a functor. Unlike the case for non-trivial sketches, a schema morphism induces three adjoint functors, the pullback and its Kan extensions. These functors can be called *data migration functors* because they transfer data from one schema to another. In this formalism, queries can be recovered as specific kinds of data migration.

Both of the above approaches give some secondary consideration to attributes, e.g. the name or salary of an employee, taking values in some data type, such as strings, integers, or booleans. Rosebrugh et al. formalized attributes in terms of infinite coproducts of a chosen terminal object, whereas Spivak formalized them by slicing the category of copresheaves over a fixed object. However, neither approach seemed to work convincingly in implementations [SW15].

1.1. The approach of this paper. In the present paper, the goal of providing a principled and workable formalization of attributes is a central concern. We consider attribute values as living in an algebra over a multi-sorted algebraic theory, capturing operations such as comparing integers or concatenating strings. A database schema is formalized as what we call an *algebraic profunctor*, which is a profunctor from a category to an algebraic theory that preserves the products of the theory. Each element of the profunctor represents an observation of a given type (string, integer, boolean) that can be made on a certain entity (employee, department). For example, if an entity has an observable for length and width, and if the theory has a multiplication, then the entity has an observable for area.

We also focus on providing syntax for algebraic databases. We can present a schema, or an instance on it, using a set of generators and relations. The generators act like the "labelled nulls" used in modern relational databases, easily handling unknown information, while the relations are able to record constraints on missing data. In this sense, our approach can be related to knowledge bases or ontologies [MS08]. One can express that Pablo is an employee whose salary is between 65 and 75, and deduce various facts; for example, if the schema expresses that each employee's salary is at most that of his or her manager, one can deduce that Pablo's manager makes at least 65.

Mathematically, this paper develops the theory of algebraic profunctors. An algebraic profunctor can be regarded as a diagram of models for an algebraic theory $\mathcal{T}$, e.g. a presheaf of rings or modules on a space. Algebraic profunctors to a fixed $\mathcal{T}$ form the objects in a

proarrow equipment — a double category satisfying a certain fibrancy condition — which we call $\mathbb{D}\mathbf{ata}$. This double category includes database schemas and schema morphisms, and we show that the horizontal morphisms (which we call *bimodules* between schemas) generalize both instances and conjunctive queries.

We make heavy use of *collages* of profunctors and bimodules. Collages are a kind of double-categorical colimit which have been studied in various guises under various names — [GS15] gives a good general treatment. We propose exactness properties which the collage construction satisfies in some examples; we say that an equipment has *extensive collages* when these properties hold. This fits in with the work started in [Sch15], and may be of interest independent of the applications in this paper. Although the present work only makes use of the properties of extensive collages in the equipment $\mathbb{P}\mathbf{rof}$ of categories, functors, and profunctors, we found more direct proofs of these properties in this case to be no easier and less illuminating.

To connect the theory with practice, it is necessary to have a concrete syntax for presenting the various categorical structures of interest. While it is mostly standard, we provide a self-contained account of a type-theoretic syntax for categories, functors, profunctors, algebraic theories, algebras over those theories, and algebraic profunctors. We use this syntax to consistently ground the theoretical development with concrete examples in the context of databases, though the reader need not have any background in that subject.

1.2. IMPLEMENTATION. The mathematical framework developed in this paper is implemented in an open-source software system we call OPL, available for download at http://categoricaldata.net/fql.html. All examples from this paper are included as built-in demonstrations in the OPL tool. We defer a detailed discussion of OPL until the end of the paper (Section 10), but two high-level introductory remarks are in order.

First, we note that most constructions on finitely-presented categories require solving word problems in categories and hence are not computable [FGR03]. Given a category presented by generators G and relations (equations) E, the word problem asks if two terms (words) in G are equal under E. Although not decidable in general, many approaches to this problem have been proposed; we discuss our particular approach in Section 10. If we can solve the word problem for a particular category presentation, then we can use that decision procedure to implement query evaluation, construct collages, and perform other tasks.

Second, we note that there are many connections between the mathematical framework presented here and various non-categorical frameworks. When restricted to a discrete algebraic theory, the query language we discuss in Section 9 corresponds exactly to relational algebra's unions of conjunctive queries under bag semantics [SW15]. This correspondence allows fragments of our framework to be efficiently implemented using existing relational systems (MySQL, Oracle, etc), and our software has indeed been used on various real-world examples [Wis+15].

1.3. OUTLINE. In Section 2 we review profunctors and use them to motivate the definition of double categories and proarrow equipments. We also review, as well as refine, the notion of collages, which exist in all of the equipments of interest in this paper. In Section 3 we review multisorted algebraic theories, and we discuss profunctors — from categories to algebraic theories — that preserve products in the appropriate way; we call these algebraic profunctors. We save relevant database-style examples until Section 4, where we provide type-theoretic syntax for presenting theories, categories and (algebraic) profunctors. This section serves as a foundation for the syntax used throughout the paper, especially in examples, though it can be skipped by those who only want to understand the category theoretic concepts.

We get to the heart of the new material in Section 5 and Section 6, where we define schemas and instances for algebraic databases and give examples. Morphisms between schemas induce three adjoint functors — called data migration functors — between their instance categories, and we discuss this in Section 7.

In Section 8 we wrap all of this into a double category (in fact a proarrow equipment) $\mathbb{D}\mathbf{ata}$, in which schemas are objects, schema morphisms are vertical morphisms, and schema bimodules — defined in this section — are horizontal morphisms. Instances are shown to be bimodules of a special sort, and the data migration functors from the previous section are shown to be obtained by composition and exponentiation of instance bimodules with representable bimodules. In this way, we see that $\mathbb{D}\mathbf{ata}$ nicely packages all of the structures and operations of interest.

Finally, in Section 9 we discuss the well-known "Select-From-Where" queries of standard database languages and show that they form a very special case of our data migration setup. We conclude with a discussion of the implementation of our mathematical framework in Section 10.

1.4. NOTATION. In this paper we will adhere to the following notation. For named categories, such as the category $\mathbf{Set}$ of sets, we use bold roman. For category variables — for instance "Let $\mathscr{C}$ be a category" — we use math script.

Named bicategories or 2-categories, such as the 2-category $\mathcal{C}\mathbf{at}$ of small categories, will be denoted similarly to named 1-categories except with calligraphic first letter. We use the same notation for a variable bicategory $\mathcal{B}$.

Double categories, such as the double category $\mathbb{P}\mathbf{rof}$ of categories, functors, and profunctors, will be denoted like 1-categories except with blackboard bold first letter. We use the same notation for a variable double category $\mathbb{D}$.

If $\mathscr{C}$ and $\mathscr{D}$ are categories, we sometimes denote the functor category $\mathbf{Cat}(\mathscr{C}, \mathscr{D})$ by $[\mathscr{C}, \mathscr{D}]$ or $\mathscr{D}^{\mathscr{C}}$.

1.5. ACKNOWLEDGEMENTS. The authors thank the anonymous referee for many helpful and questions and comments.

2. Profunctors and proarrow equipments

We begin with a review of profunctors, which are sometimes called correspondences or distributors; standard references include [Bor94a] and [Bén00]. Together with categories and functors, these fit into a proarrow equipment in the sense of Wood [Woo82; Woo85], though we follow the formulation in terms of double categories called framed bicategories (or fibrant double categories), due to Shulman [Shu08; Shu10]. Eventually, in Section 8, we will produce an equipment $\mathbb{Data}$ that encompasses database schemas, morphisms, instances, and queries.

2.1. PROFUNCTORS. Perhaps the most important example of an equipment is that of categories, functors, and profunctors. We review profunctors here, as they will be a central player in our story.

Let $\mathscr{C}$ and $\mathscr{D}$ be categories. Recall that a *profunctor* M from $\mathscr{C}$ to $\mathscr{D}$, written $M\colon \mathscr{C} \nrightarrow \mathscr{D}$, is defined to be a functor $M\colon \mathscr{C}^{\mathrm{op}} \times \mathscr{D} \to \mathbf{Set}$.

2.2. PROFUNCTORS AS MATRICES. It can be helpful to think of profunctors as something like matrices. Given finite sets X and Y, there is an equivalence between
- $X \times Y$-matrices A (i.e. functions $X \times Y \to \mathbb{R}$),
- functions $A\colon X \to \mathbb{R}^Y$,
- functions $A\colon Y \to \mathbb{R}^X$,
- linear maps $L_A\colon \mathbb{R}^X \to \mathbb{R}^Y$,
- linear maps $L'_A\colon \mathbb{R}^Y \to \mathbb{R}^X$.

Similarly, there is an equivalence between
- profunctors $M\colon \mathscr{C} \nrightarrow \mathscr{D}$,
- functors $M\colon \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}^{\mathscr{D}}$,
- functors $M\colon \mathscr{D} \to \mathbf{Set}^{\mathscr{C}^{\mathrm{op}}}$,
- colimit-preserving functors $\Lambda_M\colon \mathbf{Set}^{\mathscr{C}} \to \mathbf{Set}^{\mathscr{D}}$,
- colimit-preserving functors $\Lambda'_M\colon \mathbf{Set}^{\mathscr{D}^{\mathrm{op}}} \to \mathbf{Set}^{\mathscr{C}^{\mathrm{op}}}$.

The first three correspondences are straightforward by the cartesian monoidal closed structure of $\mathbf{Cat}$. The last two follow from the fact that, just as $\mathbb{R}^Y$ is the free real vector space on the set Y, the category $\mathbf{Set}^{\mathscr{D}^{\mathrm{op}}}$ is the free completion of $\mathscr{D}$ under colimits, and similarly for $\mathbf{Set}^{\mathscr{C}}$. By the equivalence between colimit-preserving functors $\mathbf{Set}^{\mathscr{C}} \to \mathscr{E}$ and functors $\mathscr{C}^{\mathrm{op}} \to \mathscr{E}$ for any cocomplete category $\mathscr{E}$, the functor Λ_M is obtained by taking the left Kan extension of $M\colon \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}^{\mathscr{D}}$ along the Yoneda embedding $\mathbf{y}\colon \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}^{\mathscr{C}}$. Using the pointwise formula for Kan extensions, this means that given any $I\colon \mathscr{C} \to \mathbf{Set}$, the functor $\Lambda_M(I)\colon \mathscr{D} \to \mathbf{Set}$ is given by the coend formula

$$(\Lambda_M I)(d) = \int^{c\in\mathscr{C}} I(c) \times M(c,d). \tag{1}$$

This is analogous to the matrix formula $(L_A v)_y = \sum_{x\in X} v_x A_{x,y}$.

Alternatively, since colimits in $\mathbf{Set}^{\mathscr{D}}$ are computed pointwise, we can express $\Lambda_M I$

itself as a coend in $\mathbf{Set}^{\mathscr{D}}$

$$\Lambda_M I = \int^{c\in\mathscr{C}} I(c) \cdot M(c) \tag{2}$$

where we think of M as a functor $\mathscr{C}^{\mathrm{op}} \to \mathbf{Set}^{\mathscr{D}}$. The symbol $\cdot$ represents the set-theoretic copower (see [Kel05]), i.e. $I(c)\cdot M(c)$ is an $I(c)$-fold coproduct of copies of $M(c)$. Formula (2) is analogous to the matrix formula $L_A v = \sum_{x\in X} A(x)v_x$, where we think of A as a function $X \to \mathbb{R}^Y$ and $A(x)v_x$ denotes scalar multiplication by $v_x \in \mathbb{R}$. The construction of Λ'_M is very similar.

2.3. PROFUNCTORS AS BIMODULES. One can also think of a profunctor as a sort of graded bimodule: for each pair of objects $c \in \mathscr{C}$ and $d \in \mathscr{D}$ there is a set $M(c,d)$ of elements in the bimodule, and given an element $m \in M(c,d)$ and morphisms $f\colon c' \to c$ in $\mathscr{C}$ and $g\colon d \to d'$ in $\mathscr{D}$, there are elements $g\cdot m \in M(c,d')$ and $m\cdot f \in M(c',d)$, such that the equations $(g\cdot m)\cdot f = g\cdot(m\cdot f)$, $g'\cdot(g\cdot m) = (g'\circ g)\cdot m$, and $(m\cdot f)\cdot f' = m\cdot(f\circ f')$ hold whenever they make sense.

2.4. REPRESENTABLE PROFUNCTORS. Profunctors also act as generalized functors, just like relations $R \subseteq A \times B$ act as generalized functions $A \to B$. Any functor $F\colon \mathscr{C} \to \mathscr{D}$ induces profunctors $\mathscr{D}(F,-)\colon \mathscr{C} \nrightarrow \mathscr{D}$ and $\mathscr{D}(-,F)\colon \mathscr{D} \nrightarrow \mathscr{C}$, called the profunctors *represented* by F. These profunctors are defined by

$$\mathscr{D}(F,-)(c,d) := \mathscr{D}(Fc,d) \qquad \mathscr{D}(-,F)(d,c) := \mathscr{D}(d,Fc). \tag{3}$$

2.5. TENSOR PRODUCT OF PROFUNCTORS. Given two profunctors

$$\mathscr{C} \xrightarrow{\;M\;} \mathscr{D} \xrightarrow{\;N\;} \mathscr{E}$$

there is a tensor product $M \odot N\colon \mathscr{C} \nrightarrow \mathscr{E}$, given by the coend formula

$$(M \odot N)(c,e) = \int^{d\in\mathscr{D}} M(c,d) \times N(d,e). \tag{4}$$

Following Section 2.2, this is analogous to matrix multiplication: $(AB)_{i,k} = \sum_j A_{i,j}B_{j,k}$. Equivalently, $(M \odot N)(c,e)$ is the coequalizer of the diagram

$$\coprod_{d_1,d_2\in\mathscr{D}} M(c,d_1) \times \mathscr{D}(d_1,d_2) \times N(d_2,e) \rightrightarrows \coprod_{d\in\mathscr{D}} M(c,d) \times N(d,e) \tag{5}$$

where the two maps are given by the right action of $\mathscr{D}$ on M and by the left action of $\mathscr{D}$ on N. In the notation of Section 2.3, we can write elements of $(M \odot N)(c,e)$ as tensors $m \otimes n$, where $m \in M(c,d)$ and $n \in N(d,e)$ for some $d \in \mathscr{D}$. The coequalizer then implies that $(m\cdot f)\otimes n = m\otimes(f\cdot n)$ whenever the equation makes sense. Notice the similarity to the tensor product of bimodules over rings.

Alternatively, we can define the tensor product by the composition

$$M \odot N = \mathscr{C}^{\mathrm{op}} \xrightarrow{M} \mathbf{Set}^{\mathscr{D}} \xrightarrow{\Lambda_N} \mathbf{Set}^{\mathscr{E}},$$

or by the composition $\Lambda'_N \circ M \colon \mathscr{C} \to \mathbf{Set}^{\mathscr{E}^{\mathrm{op}}}$. This is clearly equivalent to (4), using (1).

For any category $\mathscr{C}$, there is a profunctor $\mathrm{Hom}_{\mathscr{C}} \colon \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathbf{Set}$, which we will often write as $\mathscr{C} = \mathrm{Hom}_{\mathscr{C}}$ when unambiguous. For any functors $F \colon \mathscr{C} \to \mathbf{Set}$ and $G \colon \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$, there are natural isomorphisms

$$\int^{c \in \mathscr{C}} F(c) \times \mathscr{C}(c, c') \cong F(c') \qquad \int^{c \in \mathscr{C}} \mathscr{C}(c', c) \times G(c) \cong G(c'), \qquad (6)$$

a result sometimes referred to as the coYoneda lemma [Kel05, (3.71)]. Continuing with the analogy from Section 2.2, $\mathrm{Hom}_{\mathscr{C}}$ acts like an identity matrix: $\sum_i \delta_{i,j} v_i = v_j$. That is, these hom profunctors act as units for the tensor product, since (6) shows that $\mathrm{Hom}_{\mathscr{C}} \odot M \cong M \cong M \odot \mathrm{Hom}_{\mathscr{D}}$. Following Section 2.3, one can think of $\mathrm{Hom}_{\mathscr{C}}$ as the regular $(\mathscr{C}, \mathscr{C})$-bimodule, i.e. as $\mathscr{C}$ acting on itself on both sides [Mat89].

2.6. PROFUNCTOR MORPHISMS. A morphism $\phi \colon M \Rightarrow N$ between two profunctors

$$\mathscr{C} \overset{M}{\underset{N}{\rightrightarrows}} \mathscr{D},$$

is defined to be a natural transformation between the set-valued functors. In other words, for each $c \in \mathscr{C}$ and $d \in \mathscr{D}$ there is a component function $\phi_{c,d} \colon M(c, d) \to N(c, d)$ such that the equation $\phi(f \cdot m \cdot g) = f \cdot \phi(m) \cdot g$ holds whenever it makes sense.

Categories, profunctors, and profunctor morphisms form a bicategory $\mathbf{Prof}$. To explain how functors fit in, we need to discuss proarrow equipments.

2.7. PROARROW EQUIPMENTS. Before going into more properties of profunctors, it will be useful to put them in a more general and abstract framework. A *double category* is a 2-category-like structure involving two types of 1-cell—horizontal and vertical—as well as 2-cells. A *proarrow equipment* (which we typically abbreviate to just *equipment*) is a double category satisfying a certain fibrancy condition. An excellent reference is the paper [Shu08], where they are called *framed bicategories*.

We will see in Example 2.12 that there is an equipment $\mathbf{Prof}$ whose objects are categories, whose vertical 1-cells are functors, and whose horizontal 1-cells are profunctors. This is the motivating example to keep in mind for equipments. In Section 8 we will define $\mathbb{D}\mathbf{ata}$, the other main proarrow equipment of the paper, whose objects are database schemas.

2.8. DEFINITION. *A* double category $\mathbb{D}$ *consists of the following data:*
- *A category* $\mathbb{D}_0$, *which we refer to as the* vertical category *of* $\mathbb{D}$. *For any two objects* $A, B \in \mathbb{D}_0$, *we will write* $\mathbb{D}_0(A, B)$ *for the set of vertical arrows from A to B. We refer to objects of* $\mathbb{D}_0$ *as objects of* $\mathbb{D}$.

- *A category* $\mathbb{D}_1$, *equipped with two functors* $\mathsf{L}, \mathsf{R} \colon \mathbb{D}_1 \to \mathbb{D}_0$, *called the* left frame *and* right frame *functors. Given an object* $M \in \mathrm{Ob}\,\mathbb{D}_1$ *with* $A = \mathsf{L}(M)$ *and* $B = \mathsf{R}(M)$, *we say that* M *is a* proarrow *(or* horizontal arrow*) from* A *to* B *and write* $M \colon A \nrightarrow B$. *A morphism* $\phi \colon M \to N$ *in* $\mathbb{D}_1$ *is called a* 2-cell, *and is drawn as follows, where* $f = \mathsf{L}(\phi)$ *and* $g = \mathsf{R}(\phi)$:

$$
\begin{array}{ccc}
A & \xrightarrow{\;\;M\;\;} & B \\
{\scriptstyle f}\downarrow & {\Downarrow\phi} & \downarrow{\scriptstyle g} \\
C & \xrightarrow[\;\;N\;\;]{} & D
\end{array}
\tag{7}
$$

- *A* unit *functor* $\mathsf{U} \colon \mathbb{D}_0 \to \mathbb{D}_1$, *which is a section of both L and R, i.e.* $\mathsf{L} \circ \mathsf{U} = \mathrm{id}_{\mathbb{D}_0} = \mathsf{R} \circ \mathsf{U}$. *We will often write* U_A *or even A for the unit proarrow,* $\mathsf{U}(A) \colon A \nrightarrow A$, *and similarly* U_f *of just f for* $\mathsf{U}(f)$.
- *A functor* $\odot \colon \mathbb{D}_1 \times_{\mathbb{D}_0} \mathbb{D}_1 \to \mathbb{D}_1$, *called* horizontal composition, *that is weakly associative and weakly unital in the sense that there are coherent unitor and associator isomorphisms. See [Shu08] for details.*

Given a double category $\mathbb{D}$, *we will sometimes write* $\mathbf{Vert}(\mathbb{D})$ *for the vertical category* $\mathbb{D}_0$. *There is also a* horizontal bicategory, *denoted* $\mathcal{H}(\mathbb{D})$, *whose objects and 1-cells are the objects and horizontal 1-cells of* $\mathbb{D}$, *and whose 2-cells are the 2-cells of* $\mathbb{D}$ *of the form* (7) *such that* $f = \mathrm{id}_A$ *and* $g = \mathrm{id}_B$.

Given f, g, M, N *as in* (7), *we write* $_f\mathbb{D}_g(M, N)$ *for the set of 2-cells from M to N with frames f and g, and write* $\mathcal{H}(\mathbb{D})(M, N)$ *for the case where f and g are identity morphisms. If A and B are objects, then* $\mathbb{D}(A, B)$ *will always mean the set of* vertical *arrows from A to B, where* $\mathcal{H}(\mathbb{D})(A, B)$ *is used when we want the category of proarrows.*

We follow the convention of writing horizontal composition serially, i.e. the horizontal composite of proarrows $M \colon A \nrightarrow B$ and $N \colon B \nrightarrow C$, is $M \odot N \colon A \nrightarrow C$.

2.9. DEFINITION. *A double category* $\mathbb{D}$ *is* right closed *[resp.* left closed*] when its horizontal bicategory is, i.e. when composing a proarrow N [resp. M] with an arbitrary proarrow, $(- \odot N)$, [resp. $(M \odot -)$] has a left adjoint. Following [Shu08], we denote this left adjoint by $(N \rhd -)$ [resp. by $(- \lhd M)$]; hence there are bijections*

$$
\mathcal{H}(\mathbb{D})(X \odot N, P) \cong \mathcal{H}(\mathbb{D})(X, N \rhd P)
$$
$$
\mathcal{H}(\mathbb{D})(M \odot X, P) \cong \mathcal{H}(\mathbb{D})(X, P \lhd M)
$$

natural in X and P. $\mathbb{D}$ *is* biclosed *when both adjoints exist.*

Recall from [Bor94b] the definitions of cartesian morphisms and fibrations of categories.

2.10. DEFINITION. *A* proarrow equipment *(or just* equipment*) is a double category* $\mathbb{D}$ *in which the frame functor*

$$
(\mathsf{L}, \mathsf{R}) \colon \mathbb{D}_1 \to \mathbb{D}_0 \times \mathbb{D}_0
$$

is a fibration. If $f \colon A \to C$ and $g \colon B \to D$ are vertical morphisms and $N \colon C \nrightarrow D$ is a proarrow, a cartesian morphism $M \to N$ in $\mathbb{D}_1$ over (f, g) is a 2-cell

$$
\begin{array}{ccc}
A & \xrightarrow{\;M\;} & B \\
{\scriptstyle f}\big\downarrow & \Downarrow\mathrm{cart} & \big\downarrow{\scriptstyle g} \\
C & \xrightarrow[\;N\;]{} & D
\end{array}
$$

which we call a cartesian 2-cell. *We refer to M as the* restriction of N along f and g, *written $M = N(f, g)$.*

Equivalently, an equipment is a double category in which every vertical arrow $f \colon A \to B$ has a companion *$\widehat{f} \colon A \nrightarrow B$ and a* conjoint *$\widecheck{f} \colon B \nrightarrow A$, together with 2-cells satisfying certain equations (see [Shu08]). In this view, the canonical cartesian lifting of some proarrow N along (f, g) is given by $N(f, g) \cong \widehat{f} \odot N \odot \widecheck{g}$.*

2.11. ADJUNCTION BETWEEN REPRESENTABLE PROARROWS. Any vertical morphism in an equipment $\mathbb{D}$ induces an adjunction $\widehat{f} \dashv \widecheck{f}$ in the horizontal bicategory $\mathcal{H}(\mathbb{D})$, with unit denoted η_f and counit denoted ϵ_f. Moreover, the following bijective correspondences hold for any vertical morphisms $f \colon A \to B$, $g \colon C \to D$, and proarrows $M \colon A \nrightarrow B$, $N \colon C \nrightarrow D$:

$$
\begin{aligned}
{}_f\mathbb{D}_g(M, N) &\cong \mathcal{H}(\mathbb{D})(M, \widehat{f} \odot N \odot \widecheck{g}) \\
&\cong \mathcal{H}(\mathbb{D})(M \odot \widehat{g}, \widehat{f} \odot N) \\
&\cong \mathcal{H}(\mathbb{D})(\widecheck{f} \odot M, N \odot \widecheck{g}) \\
&\cong \mathcal{H}(\mathbb{D})(\widecheck{f} \odot M \odot \widehat{g}, N).
\end{aligned}
\tag{8}
$$

The last bijection shows that in an equipment, the frame functor $(\mathsf{L}, \mathsf{R}) \colon \mathbb{D}_1 \to \mathbb{D}_0 \times \mathbb{D}_0$ turns out to also be an opfibration.

We record some notation for (8). Given a 2-cell $\phi \in {}_f\mathbb{D}_g(M, N)$, we write $\widehat{\phi} \in \mathcal{H}(\mathbb{D})(M \odot \widehat{g}, \widehat{f} \odot N)$ and $\widecheck{\phi} \in \mathcal{H}(\mathbb{D})(\widecheck{f} \odot M, N \odot \widecheck{g})$ for its image under the above bijections,

$$
\begin{array}{ccc}
A & \xrightarrow{\;M\;} & B \\
{\scriptstyle f}\big\downarrow & \Downarrow\phi & \big\downarrow{\scriptstyle g} \\
C & \xrightarrow[\;N\;]{} & D
\end{array}
\qquad
\begin{array}{ccccc}
A & \xrightarrow{\;M\;} & B & \xrightarrow{\;\widehat{g}\;} & D \\
\big\| & & \Downarrow\widehat{\phi} & & \big\| \\
A & \xrightarrow[\;\widehat{f}\;]{} & C & \xrightarrow[\;N\;]{} & D
\end{array}
\qquad
\begin{array}{ccccc}
C & \xrightarrow{\;\widecheck{f}\;} & A & \xrightarrow{\;M\;} & B \\
\big\| & & \Downarrow\widecheck{\phi} & & \big\| \\
C & \xrightarrow[\;N\;]{} & D & \xrightarrow[\;\widecheck{g}\;]{} & B
\end{array}
$$

2.12. EXAMPLE. There is a double category $\mathbb{P}\mathbf{rof}$ defined as follows. The vertical category is $\mathbb{P}\mathbf{rof}_0 = \mathbf{Cat}$ the category of small categories and functors. Given objects $\mathscr{C}, \mathscr{D} \in \mathbb{P}\mathbf{rof}$, a horizontal arrow between them is a profunctor $M \colon \mathscr{C} \nrightarrow \mathscr{D}$, as described in Section 2.1. A 2-cell $\phi \in {}_F\mathbb{P}\mathbf{rof}_G(M, N)$, as to the left of (9), denotes a natural

transformation, as to the right of (9), with components $\phi_{c,d} \colon M(c,d) \to N(Fc, Gd)$:

$$
\begin{array}{ccc}
\mathscr{C} & \xrightarrow{\;M\;} & \mathscr{D} \\
{\scriptstyle F}\downarrow & \Downarrow\phi & \downarrow{\scriptstyle G} \\
\mathscr{E} & \xrightarrow[\;N\;]{} & \mathscr{F}
\end{array}
\qquad\qquad
\begin{array}{c}
\mathscr{C}^{\mathrm{op}} \times \mathscr{D} \xrightarrow{\;F^{\mathrm{op}}\times G\;} \mathscr{E}^{\mathrm{op}} \times \mathscr{F} \\
{\scriptstyle M}\searrow \quad {\overset{\phi}{\Rightarrow}} \quad \swarrow {\scriptstyle N} \\
\mathbf{Set}
\end{array}
\tag{9}
$$

The horizontal composite of profunctors $M \odot N$ is defined by the coend (4), or equivalently by the coequalizer (5), and the horizontal unit is $\mathsf{U}_{\mathscr{C}} = \mathrm{Hom}_{\mathscr{C}} \colon \mathscr{C} \nrightarrow \mathscr{C}$. This gives $\mathbb{P}\mathbf{rof}$ the structure of a double category, such that $\mathcal{H}(\mathbb{P}\mathbf{rof})$ is the bicategory $\mathcal{P}\mathbf{rof}$ defined in Section 2.6.

Moreover, the double category $\mathbb{P}\mathbf{rof}$ is biclosed (see Definition 2.9): given proarrows $M \colon \mathscr{C} \nrightarrow \mathscr{D}$, $N \colon \mathscr{D} \nrightarrow \mathscr{E}$, and $P \colon \mathscr{C} \nrightarrow \mathscr{E}$, one defines left and right exponentiation using ends

$$
(N \rhd P)(c,d) = \int_{e \in \mathscr{E}} P(c,e)^{N(d,e)} = [\mathscr{E}, \mathbf{Set}](N(d,-), P(c,-))
$$

$$
(P \lhd M)(d,e) = \int_{c \in \mathscr{C}} P(c,e)^{M(c,d)} = [\mathscr{C}^{\mathrm{op}}, \mathbf{Set}](M(-,d), P(-,e))
$$

which evidently inherit left and right actions from the respective categories when viewed as bimodules.

Finally, $\mathbb{P}\mathbf{rof}$ is an equipment because for any F, G, N as in (9), there is a cartesian 2-cell whose domain is precisely the profunctor $N(F, G) \coloneqq N \circ (F^{\mathrm{op}} \times G)$ obtained by composition. The companion and conjoint of any functor $F \colon \mathscr{C} \to \mathscr{D}$ are the representable profunctors (3)

$$
\widehat{F} = \mathscr{D}(F, -) \quad \text{and} \quad \widecheck{F} = \mathscr{D}(-, F).
$$

Thus we can also represent the cartesian lifting as $N(F, G) = \widehat{F} \odot N \odot \widecheck{G}$.

2.13. DEFINITION. *Let $\mathscr{I}$ be a small category. We say that a double category $\mathbb{D}$ has local colimits of shape $\mathscr{I}$ if, for each pair of objects $A, B \in \mathbb{D}$, the hom-category $\mathcal{H}(\mathbb{D})(A, B)$ has colimits of shape $\mathscr{I}$ and these are preserved by horizontal composition on both sides,*

$$
L \odot (\mathrm{colim}_{i \in \mathscr{I}} M_i) \cong \mathrm{colim}_{i \in \mathscr{I}} (L \odot M_i)
$$

$$
(\mathrm{colim}_{i \in \mathscr{I}} M_i) \odot N \cong \mathrm{colim}_{i \in \mathscr{I}} (M_i \odot N).
$$

We say that $\mathbb{D}$ has local colimits if it has local colimits of shape $\mathscr{I}$ for all small $\mathscr{I}$.

2.14. EXAMPLE. The equipment $\mathbb{P}\mathbf{rof}$ has local colimits. Indeed, each horizontal bicategory is a category of set-valued functors. Colimits exist, and they are preserved by horizontal composition because composition is defined by coends, which are themselves colimits.

2.15. COLLAGE OF A PROARROW. In some equipments $\mathbb{D}$, a proarrow can be represented in a certain sense by an object in $\mathbb{D}$, called its collage. For example, it is well known that a profunctor can be represented by a category, as we review in Example 2.19. In this section we collect some useful properties of the collage construction, in an arbitrary equipment.

We note briefly that the collage construction was also studied in [Woo85], in a slightly different setting. The definition we give below of an equipment with extensive collages is somewhat more general than the set of axioms considered in [Woo85], as we don't require the existence of Kleisli objects for (horizontal) monads.

2.16. DEFINITION. *Let* $M \colon A \nrightarrow B$ *be a proarrow in an equipment* $\mathbb{D}$. *Its* collage *is an object* $\widetilde{M}$ *equipped with vertical arrows* $i_A \colon A \to \widetilde{M} \leftarrow B \colon i_B$, *called the* collage inclusions, *together with a 2-cell*

$$
\begin{array}{ccc}
A & \xrightarrow{\ M\ } & B \\
{\scriptstyle i_A}\downarrow & \Downarrow\mu & \downarrow{\scriptstyle i_B} \\
\widetilde{M} & \xrightarrow[\widetilde{M}]{\ } & \widetilde{M},
\end{array}
\tag{10}
$$

that is universal *in the sense that any diagram as to the left below (a cocone under* M*) factors uniquely as to the right:*

$$
\begin{array}{ccc}
A & \xrightarrow{\ M\ } & B \\
{\scriptstyle f_A}\downarrow & \Downarrow f & \downarrow{\scriptstyle f_B} \\
X & \xrightarrow[X]{\ } & X
\end{array}
\quad = \quad
\begin{array}{ccc}
A & \xrightarrow{\ M\ } & B \\
{\scriptstyle i_A}\downarrow & \Downarrow\mu & \downarrow{\scriptstyle i_B} \\
\widetilde{M} & \xrightarrow[\widetilde{M}]{\ } & \widetilde{M} \\
{\scriptstyle \bar{f}}\downarrow & \Downarrow\bar{f} & \downarrow{\scriptstyle \bar{f}} \\
X & \xrightarrow[X]{\ } & X
\end{array}
\tag{11}
$$

2.17. REMARK. The existence of a 2-cell μ with the above universal property amounts to the existence of a left adjoint $\widetilde{(-)} \colon \mathbb{D}_1 \to \mathbb{D}_0$ to the unit functor U from Definition 2.8, since it establishes a bijection $\mathbb{D}_0(\widetilde{M}, X) \cong \mathbb{D}_1(M, \mathsf{U}_X)$. From this perspective, the universal 2-cell $\mu \colon M \Rightarrow \mathsf{U}_{\widetilde{M}}$, as in (10), is the unit of the adjunction.

2.18. DEFINITION. *An equipment* $\mathbb{D}$ *is said to* have collages *if every proarrow in* $\mathbb{D}$ *has a collage as in* (11). *By Remark 2.17,* $\mathbb{D}$ *has collages if and only if there exists a left adjoint* $\widetilde{(-)} \colon \mathbb{D}_1 \to \mathbb{D}_0$ *to the unit functor* U.

We say $\mathbb{D}$ *has normal collages if additionally the unit of the adjunction* μ *is cartesian.*

2.19. EXAMPLE. The proarrow equipment $\mathbb{Prof}$ has normal collages. The collage $\widetilde{M}$ of a profunctor $M\colon \mathscr{C}^{\mathrm{op}} \times \mathscr{D} \to \mathbf{Set}$ is a category where $\mathrm{Ob}(\widetilde{M}) := \mathrm{Ob}(\mathscr{C}) \sqcup \mathrm{Ob}(\mathscr{D})$, and

$$\widetilde{M}(x,y) = \begin{cases} \mathscr{C}(x,y) & \text{if } x \in \mathscr{C} \text{ and } y \in \mathscr{C} \\ M(x,y) & \text{if } x \in \mathscr{C} \text{ and } y \in \mathscr{D} \\ \emptyset & \text{if } x \in \mathscr{D} \text{ and } y \in \mathscr{C} \\ \mathscr{D}(x,y) & \text{if } x \in \mathscr{D} \text{ and } y \in \mathscr{D} \end{cases} \tag{12}$$

Composition in $\widetilde{M}$ is defined using composition in $\mathscr{C}$ and $\mathscr{D}$ and the functoriality of M. There are evident functors $i_{\mathscr{C}}\colon \mathscr{C} \to \widetilde{M}$ and $i_{\mathscr{D}}\colon \mathscr{D} \to \widetilde{M}$, and the 2-cell $\mu\colon M \Rightarrow \mathsf{U}_{\widetilde{M}}$ sends an element $m \in M(c,d)$ to $m \in \widetilde{M}(i_{\mathscr{C}}(c), i_{\mathscr{D}}(d)) = M(c,d)$. It is easy to see that μ is cartesian, so $\mathbb{Prof}$ has normal collages.

This construction satisfies the universal property (11). Suppose we are given $f_{\mathscr{C}}\colon \mathscr{C} \to \mathscr{X}$, $f_{\mathscr{D}}\colon \mathscr{D} \to \mathscr{X}$, and a 2-cell f as in (11). It is easy to see that the unique $\bar{f}\colon \widetilde{M} \to \mathscr{X}$ (and so $\mathsf{U}_{\bar{f}}\colon \mathsf{U}_{\widetilde{M}} \Rightarrow \mathsf{U}_{\mathscr{X}}$) that works is defined by cases, using $f_{\mathscr{C}}$ on objects and morphisms in $\mathscr{C}$, using $f_{\mathscr{D}}$ on objects and morphisms in $\mathscr{D}$, and using f on morphisms with domain in $\mathscr{C}$ and codomain in $\mathscr{D}$.

Note also that for any profunctor M as above, there is an induced functor $\widetilde{M} \to \mathbf{2}$, where $\mathbf{2} = \{0 \to 1\}$, sometimes called the *free arrow category*, is the collage of the terminal profunctor $\{*\} \nrightarrow \{*\}$. In fact, if $\mathbf{Cat}/\mathbf{2}$ denotes the slice category, it is not hard to check that the collage construction provides an equivalence of categories

$$\mathbb{Prof}_1 \simeq \mathbf{Cat}/\mathbf{2} \tag{13}$$

In particular, from a functor $F\colon \mathscr{A} \to \mathbf{2}$ we obtain a profunctor between the pullbacks of F along $0, 1\colon \{*\} \to \mathbf{2}$ respectively.

2.20. PROARROWS BETWEEN COLLAGES; SIMPLICES. We now want to consider general proarrows $\widetilde{M} \nrightarrow \widetilde{N}$ between collages in $\mathbb{D}$, by defining a category of simplices. Although we will only need this in the case $\mathbb{D} = \mathbb{Prof}$, we found the proofs simpler in the general case.

For intuition, consider two profunctors $M\colon \mathscr{C}_0 \nrightarrow \mathscr{C}_1$ and $N\colon \mathscr{D}_0 \nrightarrow \mathscr{D}_1$. A profunctor $X\colon \widetilde{M} \nrightarrow \widetilde{N}$ must assign a set $X(c,d)$ in four different cases: c is an object in either $\mathscr{C}_0$ or $\mathscr{C}_1$, and likewise for d. We could try splitting X into four profunctors $X_{i,j}\colon \mathscr{C}_i \nrightarrow \mathscr{D}_j$, but this would not encode all of the functorial actions needed to recover X. For instance, given objects $c \in \mathscr{C}_0$, $c' \in \mathscr{C}_1$, and $d \in \mathscr{D}_0$, and given an element $x \in X_{1,0}(c',d)$ and a morphism $m\colon c \to c'$ in $\widetilde{M}$ (i.e. an element $m \in M(c,c')$), there is an element $m \cdot x \in X_{0,0}(c,d)$. The idea behind the following construction is to encode all of the data of a profunctor X between collage objects by four profunctors, together with four 2-cells which capture all of those functorial actions.

2.21. DEFINITION. *Let $M\colon A_0 \nrightarrow A_1$ and $N\colon B_0 \nrightarrow B_1$ be proarrows in $\mathbb{D}$. We define an (M, N)-simplex X to be a collection of proarrows $\{X_{0,0}, X_{0,1}, X_{1,0}, X_{1,1}\}$*

$$
\begin{array}{ccc}
A_1 & \xrightarrow{\;X_{1,0}\;} & B_0 \\
M\uparrow \quad {\scriptstyle X_{0,0}} \;\;\times\;\; {\scriptstyle X_{1,1}} & & \downarrow N \\
A_0 & \xrightarrow[\;X_{0,1}\;]{} & B_1
\end{array}
$$

together with four 2-cells $X_{0,*}, X_{1,*}, X_{*,0}, X_{*,1}$ as in

$$
\begin{array}{ccc}
A_1 & \xrightarrow{X_{1,k}} & \\
M\uparrow \;\; \Downarrow X_{*,k} & & B_k \\
A_0 & \xrightarrow{X_{0,k}} &
\end{array}
\qquad
\begin{array}{ccc}
& \xrightarrow{X_{j,0}} & B_0 \\
A_j \;\; \Downarrow X_{j,*} & & \downarrow N \\
& \xrightarrow{X_{j,1}} & B_1
\end{array}
$$

such that the following equation holds:

$$
\begin{array}{ccc}
A_1 & \xrightarrow{\;X_{1,0}\;} & B_0 \\
M\uparrow \;\; {\scriptstyle\Downarrow X_{*,0}} \;\; {\scriptstyle X_{0,0}} \;\; {\scriptstyle\Downarrow X_{0,*}} & & \downarrow N \\
A_0 & \xrightarrow[\;X_{0,1}\;]{} & B_1
\end{array}
\quad = \quad
\begin{array}{ccc}
A_1 & \xrightarrow{\;X_{1,0}\;} & B_0 \\
M\uparrow \;\; {\scriptstyle\Downarrow X_{1,*}} \;\; {\scriptstyle X_{1,1}} \;\; {\scriptstyle\Downarrow X_{*,1}} & & \downarrow N \\
A_0 & \xrightarrow[\;X_{0,1}\;]{} & B_1
\end{array}
$$

A morphism $\alpha\colon X \to Y$ between two (M, N)-simplices consists of component 2-cells $\alpha = (\alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1})$, where $\alpha_{j,k}\colon X_{j,k} \to Y_{j,k}$ satisfy four evident equations. We have thus defined the *category of (M, N)-simplices*, denoted ${}_M\mathbf{Simp}_N$.

Suppose that the equipment $\mathbb{D}$ has local initial objects; see Definition 2.13. Then for any proarrow $M\colon A_0 \nrightarrow A_1$, there is an (M, M)-simplex given by the proarrows

$$
\begin{array}{ccc}
A_1 & \xrightarrow{\;0\;} & A_0 \\
M\uparrow \quad {\scriptstyle A_0} \;\;\times\;\; {\scriptstyle A_1} & & \downarrow M \\
A_0 & \xrightarrow[\;M\;]{} & A_1
\end{array}
\tag{14}
$$

together with the evident 2-cells; we call this the *unit simplex* on M and denote it by $1_M \in {}_M\mathbf{Simp}_M$.

2.22. THE FUNCTOR ${}_M\mathrm{Res}_N$. There is a functor ${}_M\mathrm{Res}_N\colon \boldsymbol{\mathcal{H}}(\mathbb{D})(\widetilde{M}, \widetilde{N}) \to {}_M\mathbf{Simp}_N$ defined as follows. On some $P\colon \widetilde{M} \nrightarrow \widetilde{N}$, the four proarrows are given by the restrictions along the collage inclusions $i_{A_j}\colon A_j \to \widetilde{M}$ and $i_{B_k}\colon B_k \to \widetilde{N}$, namely $X_{j,k} = \widehat{i}_{A_j} \odot P \odot \widecheck{i}_{B_k}$, and the 2-cells are given by horizontal composition with the universal μ_M, μ_N.

The following proposition follows directly from definitions.

2.23. PROPOSITION. *Suppose that $\mathbb{D}$ has local initial objects and collages. The four 2-cells*

$$
\begin{array}{ccc}
A \xrightarrow{\;A\;} A & \quad A \xrightarrow{\;M\;} B & \quad B \xrightarrow{\;0\;} A & \quad B \xrightarrow{\;B\;} B \\
\end{array}
$$

$$
\begin{array}{ccccccc}
i_A\downarrow \quad \Downarrow i_A \quad \downarrow i_A & \quad i_A\downarrow \quad \Downarrow \mu \quad \downarrow i_B & \quad i_B\downarrow \quad \Downarrow ! \quad \downarrow i_A & \quad i_B\downarrow \quad \Downarrow i_B \quad \downarrow i_B \\
\widetilde{M} \xrightarrow{\;\widetilde{M}\;} \widetilde{M} & \quad \widetilde{M} \xrightarrow{\;\widetilde{M}\;} \widetilde{M} & \quad \widetilde{M} \xrightarrow{\;\widetilde{M}\;} \widetilde{M} & \quad \widetilde{M} \xrightarrow{\;\widetilde{M}\;} \widetilde{M}
\end{array}
\tag{15}
$$

induce a morphism $u_M \colon 1_M \to {}_M\mathrm{Res}_M(\mathsf{U}_{\widetilde{M}})$ in ${}_M\mathbf{Simp}_M$ by unique factorization through cartesian 2-cells. The following are equivalent

1. *u_M is an isomorphism in ${}_M\mathbf{Simp}_M$.*
2. *each of the four squares in (15) is cartesian.*
3. *the four induced 2-cells are isomorphisms:*

$$
\eta_{i_A} \colon \mathsf{U}_A \xrightarrow{\;\sim\;} \widehat{i}_A \odot \widecheck{i}_A, \quad \mu \colon M \xrightarrow{\;\sim\;} \widehat{i}_A \odot \widecheck{i}_B, \quad ! \colon 0 \xrightarrow{\;\sim\;} \widehat{i}_B \odot \widecheck{i}_A, \quad \eta_{i_B} \colon \mathsf{U}_B \xrightarrow{\;\sim\;} \widehat{i}_B \odot \widecheck{i}_B.
\tag{16}
$$

Note that if $\mathbb{D}$ satisfies the equivalent conditions in Proposition 2.23 then, in particular, it has normal collages.

2.24. DEFINITION. *Let $\mathbb{D}$ be an equipment. We will say that $\mathbb{D}$ has* extensive collages *if it satisfies the following conditions:*

1. *$\mathbb{D}$ has collages and local initial objects,*
2. *any of the equivalent conditions from Proposition 2.23 are satisfied,*
3. *for every pair of proarrows M and N, the functor ${}_M\mathrm{Res}_N \colon \mathcal{H}(\mathbb{D})(\widetilde{M}, \widetilde{N}) \to {}_M\mathbf{Simp}_N$ is an equivalence of categories.*

Extensive collages are best behaved in the presence of local finite colimits. The following proposition provides a condition which is equivalent to condition 3 above in this case, but which is often easier to verify. The proof provides an explicit construction of the inverse of ${}_M\mathrm{Res}_N$ using colimits in the horizontal bicategories.

2.25. PROPOSITION. *Suppose that $\mathbb{D}$ is an equipment with collages, that it satisfies condition 2 in Definition 2.24, and that $\mathbb{D}$ has local finite colimits (so it also satisfies condition 1). Then condition 3 is equivalent to the following condition:*

3'. *for any proarrow $M \colon A \nrightarrow B$, the following square is a pushout in $\mathcal{H}(\mathbb{D})(\widetilde{M}, \widetilde{M})$:*

$$
\begin{array}{ccc}
\widecheck{i}_A \odot \widehat{i}_A \odot \widecheck{i}_B \odot \widehat{i}_B & \xrightarrow{\;\epsilon_{i_A} \odot \widecheck{i}_B \odot \widehat{i}_B\;} & \widecheck{i}_B \odot \widehat{i}_B \\
{\scriptstyle \widecheck{i}_A \odot \widehat{i}_A \odot \epsilon_{i_B}}\Big\downarrow & & \Big\downarrow{\scriptstyle \epsilon_{i_B}} \\
\widecheck{i}_A \odot \widehat{i}_A & \xrightarrow[\;\epsilon_{i_A}\;]{} & \mathsf{U}_{\widetilde{M}}
\end{array}
\tag{17}
$$

PROOF. Suppose $\mathbb{D}$ has local finite colimits and satisfies condition 2. First assuming condition 3 we will show that (17) is a pushout. It suffices that its image under the equivalence $_M\mathrm{Res}_N$ (Section 2.22) is a pushout, i.e. each of the four restriction functors,

$$\widehat{i}_A \odot - \odot \widecheck{i}_A \colon \boldsymbol{\mathcal{H}}(\mathbb{D})(\widetilde{M}, \widetilde{M}) \to \boldsymbol{\mathcal{H}}(\mathbb{D})(A, A),$$

as well as $\widehat{i}_A \odot - \odot \widecheck{i}_B$, $\widehat{i}_B \odot - \odot \widecheck{i}_A$, and $\widehat{i}_B \odot - \odot \widecheck{i}_B$, take the diagram (17) to a pushout square. This follows easily from condition 2, in particular the four isomorphisms of (16).

Conversely, assuming condition 3', we will show that $_M\mathrm{Res}_N$ is an equivalence of categories for any pair of proarrows $M \colon A_0 \nrightarrow A_1$, $N \colon B_0 \nrightarrow B_1$. To define the inverse functor, let $X \in {}_M\mathbf{Simp}_N$ be a simplex, and consider the diagram
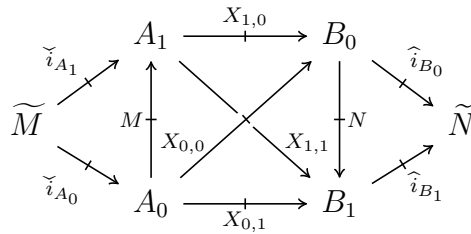
which also contains six 2-cells:

$$X_{*,k} \colon M \odot X_{1,k} \to X_{0,k}, \quad X_{j,*} \colon X_{j,0} \odot N \to X_{j,1},$$
$$\widecheck{\mu}_M \colon \widecheck{i}_{A_0} \odot M \to \widecheck{i}_{A_1}, \quad \widehat{\mu}_N \colon N \odot \widehat{i}_{B_1} \to \widehat{i}_{B_0}$$

where the μ's are universal 2-cells and $\widehat{\mu}$ and $\widecheck{\mu}$ are as in Section 2.11.

The inverse to $_M\mathrm{Res}_N$, which we denote $(X \mapsto \widetilde{X}) \colon {}_M\mathbf{Simp}_N \to \boldsymbol{\mathcal{H}}(\mathbb{D})(\widetilde{M}, \widetilde{N})$, is given by sending the simplex X to the colimit in $\boldsymbol{\mathcal{H}}(\mathbb{D})(\widetilde{M}, \widetilde{N})$ of the 3×3 square: [1]

Note that this colimit can be formed by first taking the pushout of each row, and then taking the pushout of the resulting span, or by taking column-wise pushouts first. For the time being, ignore the separated right-hand column and bottom row of (18).

---

[1]We suppress the $\odot$ symbol in the objects to reduce the required space.

We now show that $_M\mathrm{Res}_N$ and $X \mapsto \widetilde{X}$ are inverse equivalences. Suppose $P \colon \widetilde{M} \nrightarrow \widetilde{N}$ is a proarrow and let $X = {_M\mathrm{Res}_N}(P)$; we want to show that there is a natural isomorphism $P \cong \widetilde{X}$. Performing the substitution $X_{j,k} = \widehat{i}_{A_j} P \widecheck{i}_{B_k}$ and using the isomorphisms from (16), e.g. $M \cong \widehat{i}_{A_0} \widecheck{i}_{A_1}$, each row (resp. each column) can be seen as a composition of some proarrow — namely the one in the right-hand column (resp. bottom row) — with the diagram (17). Since local colimits commute with proarrow composition, the right-hand column (resp. bottom row) proarrows are indeed the pushouts. In the same way, one checks that P is the colimit of both the right-hand column and the bottom row.

In the other direction, if $X \in {_M\mathbf{Simp}_N}$ is any simplex and $\widetilde{X}$ is the colimit of the square in (18), we want to show that $_M\mathrm{Res}_N(\widetilde{X}) \cong X$. It is straightforward to check that $\widehat{i}_{A_j} \odot \widetilde{X} \odot \widecheck{i}_{B_k} \cong X_{j,k}$ by composing the square with $\widehat{i}_{A_j}$ on the left and $\widecheck{i}_{B_k}$ on the right and applying the equations of (16). It is moreover easy to see that these isomorphisms form the components of an isomorphism of simplices $_M\mathrm{Res}_N(\widetilde{X}) \cong X$. Thus $_M\mathrm{Res}_N$ is an equivalence of categories. ∎

2.26. REMARK. It is likely possible to characterize equipments with extensive collages (assuming local finite colimits) in terms of an adjunction of double categories. We won't pursue this further here, but for the interested reader we provide a rough sketch as a starting point for further investigation.

If $\mathbb{D}$ is an equipment with local finite colimits, one can define an equipment $\mathbf{Simp}(\mathbb{D})$ whose vertical category is $\mathbb{D}_1$ and whose horizontal 1-cells are simplices. The composition in $\mathbf{Simp}(\mathbb{D})$ is given by (51). There is a double functor $\mathsf{U} \colon \mathbb{D} \to \mathbf{Simp}(\mathbb{D})$ sending each object $A \in \mathbb{D}$ to the unit proarrow U_A and each proarrow $M \colon A \nrightarrow B$ to the unit simplex 1_M defined in (14).

If $\mathbb{D}$ has extensive collages, then U has a left adjoint $\mathbf{Col}$ sending each proarrow $M \in \mathbf{Simp}(\mathbb{D})$ to its collage $\mathbf{Col}(\mathbf{M})$ and acting on simplices by the pushout (18). Looking at the definition Definition 2.24, it seems that condition 1 is related to the existence of a left adjoint to U, condition 2 is related to the property that the 2-cell components of the unit of this double-adjunction are cartesian, and condition 3' is related to the property that the right adjoint $\mathbf{Col}$ is normal (preserves unit proarrows). Perhaps this observation can be worked into an equivalent characterization of equipments with extensive collages, but we leave it to the motivated reader to investigate further.

2.27. EXAMPLE. The equipment $\mathbb{P}\mathbf{rof}$ has extensive collages. Indeed, $\mathbb{P}\mathbf{rof}$ has local colimits by Example 2.14 and normal collages by Example 2.19. Moreover, we will verify that $\mathbb{P}\mathbf{rof}$ satisfies condition 3' of Proposition 2.25.

If $M \colon \mathscr{C} \nrightarrow \mathscr{D}$ is a profunctor, then we need to show that (17) is a pushout in the category $[\widetilde{M}^{\mathrm{op}} \times \widetilde{M}, \mathbf{Set}]$. It suffices to show that it is a pointwise pushout. For any objects $x, y \in \widetilde{M}$, it is not hard to see that (17) becomes one of the following pushout

squares in **Set**:

$$
\begin{array}{cc}
& y \in \mathscr{C} \qquad\qquad\qquad y \in \mathscr{D} \\
\end{array}
$$

$$
x \in \mathscr{C} \quad
\begin{array}{ccc}
0 & \longrightarrow & 0 \\
\downarrow & \ulcorner & \downarrow \\
\mathscr{C}(x,y) & \longrightarrow & \mathscr{C}(x,y)
\end{array}
\qquad
\begin{array}{ccc}
M(x,y) & \rightarrow & M(x,y) \\
\downarrow & \ulcorner & \downarrow \\
M(x,y) & \rightarrow & M(x,y)
\end{array}
$$

$$
x \in \mathscr{D} \quad
\begin{array}{ccc}
0 & \longrightarrow & 0 \\
\downarrow & \ulcorner & \downarrow \\
0 & \longrightarrow & 0
\end{array}
\qquad
\begin{array}{ccc}
0 & \longrightarrow & \mathscr{D}(x,y) \\
\downarrow & \ulcorner & \downarrow \\
0 & \longrightarrow & \mathscr{D}(x,y)
\end{array}
$$

2.28. COLLAGES AS LAX (CO)LIMITS. When an equipment $\mathbb{D}$ has extensive collages and local finite colimits (like $\mathbb{P}\mathbf{rof}$), there is another universal property involving collages, which can be expressed entirely in terms of the horizontal bicategory $\mathcal{H}(\mathbb{D})$.

2.29. DEFINITION. *Let $\mathcal{B}$ be a bicategory, let $F\colon A \to B$ be a 1-cell in $\mathcal{B}$, and let X be an object in $\mathcal{B}$. Define a category of* lax cocones *from F to X, written $_F\mathbf{Cocone}_X$, as follows: an object of $_F\mathbf{Cocone}_X$ is a diagram*

$$
\begin{array}{ccc}
B & \xrightarrow{P_B} & \\
F \uparrow \quad {\Downarrow \pi} & & X \\
A & \xrightarrow{P_A} &
\end{array}
$$

and a morphism $\alpha\colon (P_A, P_B, \pi) \to (Q_A, Q_B, \chi)$ is a pair of 2-cells $\alpha_A\colon P_A \to Q_A$ and $\alpha_B\colon P_B \to Q_B$ making an evident diagram commute.

Any cocone $(P_A, P_B, \pi) \in {_F\mathbf{Cocone}_X}$ induces a functor $\mathcal{B}(X, Y) \to {_F\mathbf{Cocone}_Y}$ by composition. If this functor is an equivalence of categories, then we say that X is a lax colimit *of the arrow F (see for example [Kel89]). Dually, there is a category $_X\mathbf{Cone}_F$ of* lax cones *from X to F, employed in the definition of* lax limits *of arrows.*

2.30. PROPOSITION. *Let $\mathbb{D}$ be an equipment with extensive collages and local finite colimits, and let $M\colon A \nrightarrow B$ be a proarrow with collage $i_A\colon A \to \widetilde{M} \leftarrow B \colon i_B$. The triangle on the left exhibits $\widetilde{M}$ as a lax colimit of the 1-cell M in $\mathcal{H}(\mathbb{D})$, and the triangle on the right exhibits $\widetilde{M}$ as a lax limit of M.*

$$
\begin{array}{ccc}
B & \xrightarrow{\widehat{i}_B} & \\
M \uparrow \quad {\Downarrow \widehat{\mu}} & & \widetilde{M} \\
A & \xrightarrow{\widehat{i}_A} &
\end{array}
\qquad\qquad
\begin{array}{ccc}
& \xrightarrow{\widecheck{i}_A} & A \\
\widetilde{M} \quad {\Downarrow \widecheck{\mu}} & & \downarrow M \\
& \xrightarrow{\widecheck{i}_B} & B
\end{array}
$$

PROOF. The 2-cells $\widehat{\mu}$, $\widecheck{\mu}$ correspond to the cartesian μ as in Section 2.11. We will show that the triangle on the left is a lax colimit cocone, i.e. that composing with $\widehat{\mu}$ induces an equivalence of categories $\mathcal{H}(\mathbb{D})(\widetilde{M}, Y) \to {}_M\mathbf{Cocone}_Y$ for any Y. We define the inverse functor to send a cocone (P_A, P_B, π) to the proarrow $P\colon \widetilde{M} \nrightarrow Y$ defined by a pushout in $\mathcal{H}(\mathbb{D})(\widetilde{M}, Y)$:

$$
\begin{array}{ccc}
\widecheck{i}_A \odot M \odot P_B & \xrightarrow{\widecheck{\mu} \odot P_B} & \widecheck{i}_B \odot P_B \\
{\scriptstyle \widecheck{i}_A \odot \pi}\downarrow & & \downarrow \\
\widecheck{i}_A \odot P_A & \xrightarrow{\quad\ulcorner\quad} & P
\end{array}
\tag{19}
$$

Suppose we start with an arbitrary proarrow $Q\colon \widetilde{M} \nrightarrow Y$, and compose with $\widehat{\mu}$ to get the cocone $\pi = \widehat{\mu} \odot Q\colon M \odot \widehat{i}_B \odot Q \to \widehat{i}_A \odot Q$. We can see that the pushout (19) is just (17) composed by Q on the right, showing $P \cong Q$. On the other hand, if we start with an arbitrary cocone π, take the pushout P as in (19), then compose on the left with $\widehat{\mu}\colon M \odot \widehat{i}_B \to \widehat{i}_A$, it is easy to check that we get π back.

Thus the pushout (19) does define an inverse functor ${}_M\mathbf{Cocone}_Y \to \mathcal{H}(\mathbb{D})(\widetilde{M}, Y)$, showing that the triangle on the left is a lax colimit cocone. The lax limit cone follows by a dual argument. ∎

2.31. REMARK. A converse to Proposition 2.30 holds: if $\mathbb{D}$ has local finite colimits such that the conclusion to Proposition 2.30 holds for all proarrows $M\colon A \nrightarrow B$ in $\mathbb{D}$, then $\mathbb{D}$ has extensive collages. We won't need this converse, and so do not prove it. The proof is straightforward, regarding a simplex as a "lax cocone of lax cones" (or visa-versa).

2.32. REMARK. For convenience, we will break down the universal property of $\widetilde{M}$ as the lax limit of M. Suppose $\mathbb{D}$ has extensive collages.

Given any $P_A\colon X \nrightarrow A$, $P_B\colon X \nrightarrow B$, and 2-cell $\pi\colon P_A \odot M \to P_B$, there is a proarrow $P\colon X \nrightarrow \widetilde{M}$ (which is unique up to isomorphism by the 2-dimensional part of the universal property of Proposition 2.30) such that $\pi \cong P \odot \widecheck{\mu}$. Namely cartesian 2-cells exist, by $P_A \cong P \odot \widecheck{i}_A$, $P_B \cong P \odot \widecheck{i}_B$, satisfying the equation (where μ is also cartesian)

$$
\begin{array}{ccc}
X \xrightarrow{P_A} A \xrightarrow{M} B & & X \xrightarrow{P_A} A \xrightarrow{M} B \\
\parallel \quad {\scriptstyle\text{cart}} \quad \downarrow{\scriptstyle i_A} \ {\scriptstyle\Downarrow\mu}\ \downarrow{\scriptstyle i_B} & = & \parallel \quad {\scriptstyle\Downarrow\pi}\quad \downarrow{\scriptstyle i_B} \\
X \xrightarrow[P]{} \widetilde{M} \xrightarrow[\widetilde{M}]{} \widetilde{M} & & X \xrightarrow[P]{} \widetilde{M}
\end{array}
\tag{20}
$$

The 2-dimensional part of the universal property says that, given $\alpha_A\colon p_A \to q_A$ and $\alpha_B\colon p_B \to q_B$ such that $\alpha_B \circ p = q \circ \alpha_A$, there is a unique $\alpha\colon P \to Q$ making the evident diagrams commute.

The universal property for the lax colimit is dual.

3. Algebraic theories

In this section, we recall some basic aspects of the well-known work on algebraic theories and their algebras [ARV11] relevant to our purposes. In particular, algebraic theories are often used to define data types within various programming languages [Mit96], and as stated in the introduction, our main goal is to connect databases and programming languages.

3.1. DEFINITION. *A (*multisorted*) algebraic theory is a cartesian strict monoidal category* $\mathcal{T}$ *together with a set* $S_{\mathcal{T}}$*, elements of which are called* base sorts*, such that the monoid of objects of* $\mathcal{T}$ *is free on* $S_{\mathcal{T}}$*. The terminal object in* $\mathcal{T}$ *is denoted* $\mathbf{1}$*.*

The category **ATh** *has algebraic theories as objects, and morphisms* $\mathcal{T} \to \mathcal{T}'$ *are product preserving functors* F *which send base sorts to base sorts: for any* $s \in S_{\mathcal{T}}$*,* $F(s) \in S_{\mathcal{T}'}$*.*

3.2. REMARK. Throughout this paper we will discuss algebraic theories—categories with finite products and functors that preserve them—which are closely related to the notion of finite product sketches; see [BW85]. However, aside from issues of syntax and computation, everything we say in this paper would also hold if algebraic theories were replaced by *essentially algebraic theories*—categories with finite limits and functors that preserve them—which are analogous to finite limit sketches.

3.3. DEFINITION. *Let* $\mathcal{T}$ *be an algebraic theory. An* algebra *(sometimes called a* model*) of* $\mathcal{T}$ *is a finite product-preserving functor* $\mathcal{T} \to \mathbf{Set}$*. The category* $\mathcal{T}$**-Alg** *of* $\mathcal{T}$*-algebras is the full subcategory of* $[\mathcal{T}, \mathbf{Set}]$ *spanned by the finite product-preserving functors.*

3.4. EXAMPLE. If $\mathcal{T}$ is an algebraic theory, and $t \in \mathcal{T}$ is an object, then the representable functor $\mathcal{T}(t, -)$ preserves finite products. Thus the Yoneda embedding $\mathbf{y} \colon \mathcal{T}^{\mathrm{op}} \to [\mathcal{T}, \mathbf{Set}]$ factors through $\mathcal{T}$**-Alg**.

In particular, $\mathbf{y}(\mathbf{1}) = \mathcal{T}(\mathbf{1}, -)$ is the initial $\mathcal{T}$-algebra for any algebraic theory, called the *algebra of constants* and denoted by $\kappa := \mathbf{y}(\mathbf{1})$.

We state the following theorem for future reference; proofs can be found in [AR94].

3.5. THEOREM. *Let* $\mathcal{T}$ *be any algebraic theory.*
- *The Yoneda embedding* $\mathbf{y} \colon \mathcal{T}^{\mathrm{op}} \to \mathcal{T}$**-Alg** *is dense. (By definition,* $\mathcal{T}$**-Alg** *is a full subcategory of* $[\mathcal{T}, \mathbf{Set}]$*.)*
- $\mathcal{T}$**-Alg** *is closed in* $[\mathcal{T}, \mathbf{Set}]$ *under sifted colimits. ([AR94, Prop. 2.5].)*
- $\mathcal{T}$**-Alg** *has all colimits. ([AR94, Thm. 4.5].)*

3.6. WARNING. Note that the forgetful functor $\mathcal{T}$**-Alg** $\to [\mathcal{T}, \mathbf{Set}]$ in general does *not* preserve colimits; i.e. colimits in $\mathcal{T}$**-Alg** are not taken pointwise. However, see Remark 6.9.

3.7. REMARK. For convenience, we will recall the notion of a dense functor, though we only use it in the case of the inclusion of a full subcategory. A functor $F \colon \mathscr{A} \to \mathscr{C}$ is *dense* if one of the following equivalent conditions holds:
- for any object $C \in \mathscr{C}$, the canonical cocone from the canonical diagram $(F \downarrow C) \to \mathscr{C}$ to C is a colimit cocone,

- the identity functor $\mathrm{id}_{\mathscr{C}}$ is the pointwise left Kan extension of F along itself,
- the representable functor $\mathscr{C}(F, -): \mathscr{C} \to [\mathscr{A}^{\mathrm{op}}, \mathbf{Set}]$ is fully faithful,
- (assuming $\mathscr{C}$ is cocomplete) for any object $C \in \mathscr{C}$, the canonical morphism $\int^{A \in \mathscr{A}} \mathscr{C}(F(A), C) \cdot F(A) \to C$ is an isomorphism.

3.8. ALGEBRAIC PROFUNCTORS. In the previous section, we recalled the basic elements of the theory of profunctors (see Sections 2.1 to 2.6). At this point, we wish to characterize those profunctors between a category and an algebraic theory $M: \mathscr{C} \nrightarrow \mathscr{T}$, which interact nicely with the products in $\mathscr{T}$.

The following equivalences are easy to establish, by translating a product-preserving condition for $M: \mathscr{C}^{\mathrm{op}} \times \mathscr{T} \to \mathbf{Set}$ under $(- \times \mathscr{A}) \dashv (-)^{\mathscr{A}}$, and by (12) for the collage construction in $\mathbb{P}\mathbf{rof}$.

3.9. LEMMA. *Let $\mathscr{C}$ be a category and $\mathscr{T}$ an algebraic theory. For any profunctor $M: \mathscr{C} \nrightarrow \mathscr{T}$, the following are equivalent:*
- *for each $c \in \mathscr{C}$, the functor $M(c, -): \mathscr{T} \to \mathbf{Set}$ preserves finite products,*
- *$M: \mathscr{T} \to \mathbf{Set}^{\mathscr{C}^{\mathrm{op}}}$ preserves finite products,*
- *$M: \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}^{\mathscr{T}}$ factors through the full subcategory $\mathscr{T}\text{-}\mathbf{Alg}$,*
- *the inclusion $i_{\mathscr{T}}: \mathscr{T} \to \widetilde{M}$ into the collage of M preserves finite products.*

3.10. DEFINITION. *We refer to a profunctor M satisfying any of the equivalent conditions of Lemma 3.9 as an* algebraic profunctor, *or we say that it* preserves products on the right. *We denote a profunctor $M: \mathscr{C} \nrightarrow \mathscr{T}$ which is algebraic, using a differently-decorated arrow*

$$M: \mathscr{C} \twoheadrightarrow \mathscr{T}.$$

We define the category $\mathbf{Prof}^{\times}$ to be the full subcategory of the pullback

$$
\begin{array}{ccc}
\mathbf{Prof}^{\times} \longrightarrow & \cdot \xrightarrow{\quad\quad} & \mathbb{P}\mathbf{rof}_1 \\
\downarrow & {}^{\lrcorner} & \downarrow {\scriptstyle (L,R)} \\
\mathbf{Cat} \times \mathbf{ATh} \xrightarrow{\quad} & \mathbf{Cat} \times \mathbf{Cat}
\end{array}
$$

spanned by the algebraic profunctors. Here, L and R are the frame functors (Definition 2.8).

Suppose given a pair of composable profunctors $\mathscr{C} \xrightarrow{M} \mathscr{D} \xrightarrow{N} \mathscr{T}$ in which the latter is algebraic. We want to compose them in such a way that the composition is also algebraic. It is not hard to see that ordinary profunctor composition $M \odot N$ does not generally satisfy this property; however, we can define a composition which does. In Definition 3.11

we will formalize this as a left action $\otimes$ of $\mathbb{P}\mathbf{rof}$ on $\mathbf{Prof}^\times$:

$$
\begin{array}{ccc}
\mathbf{Cat} & \xleftarrow{\ \ L\ \ } \mathbf{Prof}^\times \xrightarrow{\ \ R\ \ } & \mathbf{ATh} \\
\ \uparrow{\scriptstyle R} & \uparrow \qquad\qquad & \uparrow{\scriptstyle R} \\
\mathbb{P}\mathbf{rof}_1 & \xleftarrow{\ \ulcorner\ } \cdot \qquad\quad & \\
\ \downarrow{\scriptstyle L} & \ \ \searrow{\scriptstyle \otimes} & \\
\mathbf{Cat} & \xleftarrow{\qquad\qquad L\qquad\qquad} & \mathbf{Prof}^\times
\end{array}
\qquad (21)
$$

We thus aim to define a functor $\otimes$ (dotted line) from the category of composable profunctor pairs where the second is algebraic, such that the above diagram commutes.

Let $\mathcal{D}$ be a category, $\mathcal{T}$ an algebraic theory, and $N\colon \mathcal{D} \nrightarrow \mathcal{T}$ an algebraic profunctor. By Lemma 3.9, we can consider N to be a functor $N\colon \mathcal{D}^{\mathrm{op}} \to \mathcal{T}\text{-}\mathbf{Alg}$. Define the functor $\Lambda_N^\times\colon \mathbf{Set}^{\mathcal{D}} \to \mathcal{T}\text{-}\mathbf{Alg}$ by the coend formula

$$
\Lambda_N^\times(J) = \int^{d\in\mathcal{D}} J(d) \cdot N(d)
$$

taken in the category $\mathcal{T}\text{-}\mathbf{Alg}$. This coend exists because $\mathcal{T}\text{-}\mathbf{Alg}$ is cocomplete, and the formula coincides with (2), except there the coend is taken in $\mathbf{Set}^{\mathcal{T}}$, thus is pointwise.

3.11. DEFINITION. *Let $M \in \mathbf{Prof}_1(\mathcal{C}, \mathcal{D})$ be a profunctor, and let $N \in \mathbf{Prof}^\times(\mathcal{D}, \mathcal{T})$ be an algebraic profunctor. The* left tensor *of M on N, denoted $M \otimes N \in \mathbf{Prof}^\times(\mathcal{C}, \mathcal{T})$ is defined by the composition $\Lambda_N^\times \circ M\colon \mathcal{C}^{\mathrm{op}} \to \mathcal{T}\text{-}\mathbf{Alg}$.*

This left tensor can evidently be extended to a functor $\otimes$ as in (21). It is also simple to check that it defines a left action of $\mathbb{P}\mathbf{rof}$ on $\mathbf{Prof}^\times$, in the sense that $\otimes$ respects units and composition in $\mathbb{P}\mathbf{rof}$.

4. Presentations and syntax

In this section we will introduce syntax for algebraic theories, as well as for categories and (co)presheaves. In general, a presentation of a given mathematical object consists of generators and relations in a specified form. The object itself is then obtained by recursively generating *terms* according to a syntax, and then quotienting by the relations.

The material in this section is relatively standard (see, e.g. [Jac99] or [Mit96]). We go through it carefully in order to fix the notation we will use in examples.

4.1. PRESENTATIONS OF ALGEBRAIC THEORIES. The presentation of an algebraic theory, as defined in Definition 3.1, does not explicitly mention products. Instead, it relies on multi-arity function symbols on the base sorts. A signature simply lays out these sorts and function symbols.

4.2. DEFINITION. *An* algebraic signature *is a pair* $\Sigma = (S_\Sigma, \Phi_\Sigma)$*, where* S_Σ *is a set of base sorts and* Φ_Σ *is a set of* function symbols. *Each function symbol* $f \in \Phi$ *is assigned a (possibly empty, ordered) list of sorts* $\mathrm{dom}(f)$ *and a single sort* $\mathrm{cod}(f)$*. We use the notation* $f\colon (s_1, \ldots, s_n) \to s'$ *to mean that* $\mathrm{dom}(f) = (s_1, \ldots, s_n)$ *and* $\mathrm{cod}(f) = s'$*. We call* n *the* arity *of* f*; if* $n = 0$*, we say it is 0-ary and write it* $f\colon () \to s'$*.*

4.3. DEFINITION. *Let* **ASig** *denote the category of algebraic signatures. A morphism* $F\colon \Sigma \to \Sigma'$ *between signatures is a pair of functions* $F_S\colon S_\Sigma \to S_{\Sigma'}$ *and* $F_\Phi\colon \Phi_\Sigma \to \Phi_{\Sigma'}$*, such that for any function symbol* $f \in \Phi_\Sigma$ *with* $f\colon (s_1, \ldots, s_n) \to s'$*,* $\mathrm{dom}(F_\Phi f) = (F_S(s_1), \ldots, F_S(s_n))$ *and* $\mathrm{cod}(F_\Phi f) = F_S(s')$*.*

4.4. EXAMPLE. Consider the signature Σ for the algebraic theory of monoid actions on a set. There are two sorts, $S = \{m, s\}$, and three function symbols, $\eta\colon () \to m$ for the unit, $\mu\colon (m, m) \to m$ for the multiplication, and $\alpha\colon (m, s) \to s$ for the action. If Σ' is the signature for the theory of monoids, there is an evident inclusion morphism $\Sigma' \to \Sigma$.

4.5. EXAMPLE. Every algebraic theory $\mathscr{T}$ has an underlying algebraic signature $\Sigma_{\mathscr{T}}$, whose base sorts are those of $\mathscr{T}$, and whose function symbols $f\colon (s_1, \ldots, s_n) \to s'$ are the morphisms $f \in \mathscr{T}(s_1 \times \cdots \times s_n, s')$. This defines a functor $U\colon \textbf{ATh} \to \textbf{ASig}$.

We will see in Remark 4.14 that U has a left adjoint, giving the free algebraic theory generated by a signature. We construct this left adjoint *syntactically*, and we will make use of this syntax throughout the paper.

4.6. DEFINITION. *Fix an algebraic signature* Σ*. A* context Γ *over* Σ *is formally a set* Γ_v *together with a function* $\Gamma_s\colon \Gamma_v \to S_\Sigma$*. In other words, a context is an object of the slice category* $\textbf{Set}_{/S_\Sigma}$*, or equivalently the functor category* $\textbf{Set}^{S_\Sigma}$*, regarding* S_Σ *as a discrete category. When the set* Γ_v *is finite, we will encode both* Γ_v *and* Γ_s *as a list* $\Gamma = (x_1 : s_1, \ldots, x_n : s_n)$*, and refer to* Γ *as a* finite *context.*

If $\Gamma = (x_1 : s_1, \ldots, x_n : s_n)$ *and* $\Gamma' = (x'_1 : s'_1, \ldots, x'_m : s'_m)$ *are two contexts, we will write* $\Gamma, \Gamma' = (x_1 : s_1, \ldots, x_n : s_n, x'_1 : s'_1, \ldots, x'_m : s'_m)$ *for their concatenation, equivalently given by the induced function* $\Gamma_v \sqcup \Gamma'_v \to S_\Sigma$*. In practice, when concatenating contexts, we implicitly assume that variables are renamed as necessary to avoid name clashes. We denote the empty context by* $\varnothing$*.*

4.7. REMARK. Intuitively, a context $(x_1 : s_1, \ldots, x_n : s_n)$ represents the declaration that symbol x_i "belongs to the sort" s_i. We treat the parentheses around a context as optional, and use them only as an aid to readability.

The primary role of contexts is to explicitly list the "free variables" which are allowed to be used inside an expression. Thus a context $(x : \mathsf{Int}, y : A)$ roughly corresponds to the English "let x be an integer and let y be an element of A". The next definition makes this intuition precise.

4.8. DEFINITION. *Fix an algebraic signature Σ and a context Γ. A term in context Γ is an expression built out of the variables in Γ and function symbols in Σ. Every term has an associated sort. We use the notation $\Gamma \vdash t : s$ to denote that t is a term in context Γ and that t has sort s.*

Terms in context Γ are defined recursively as follows:

- *if $(x : s) \in \Gamma$, then $\Gamma \vdash x : s$,*
- *if $f : (s_1, \ldots, s_n) \to s'$ is a function symbol in Σ and $\Gamma \vdash t_i : s_i$ for each $1 \leq i \leq n$, then $\Gamma \vdash f(t_1, \ldots, t_n) : s'$.*

We will sometimes refer to terms $\varnothing \vdash t$ in the empty context as *ground terms*. A ground term t must not contain any variables, and so must be constructed entirely out of function symbols in Σ (which includes 0-ary function symbols). Note that there can be terms in non-empty contexts which contain no variables, but we will not call these ground terms.

4.9. EXAMPLE. In Example 4.4 we gave the signature Σ for monoid actions. An example term is $x_1 : m,\ x_2 : m,\ p : s \vdash \alpha(\mu(x_1, x_2), p) : s$. An example ground term is $\varnothing \vdash \mu(\eta, \mu(\eta, \eta)) : m$.

One can think of a variable x which appears in a term t as a placeholder which can be replaced by other expressions. For instance, in $x^3 - 2x$, the variable x can be replaced by any number, or even another polynomial. To make this precise, the operation of substitution is defined recursively.

4.10. DEFINITION. *Let Θ, Γ, and Ψ be contexts. If $(\Theta, x : s, \Psi) \vdash t : s'$ and $\Gamma \vdash u : s$ are terms, then $\Theta, \Gamma, \Psi \vdash t[x := u] : s'$ denotes the term obtained by replacing all occurrences of x in t with u. This substitution operation is defined formally by recursion:*

- $x[x := u] = u$,
- $x'[x := u] = x'$ *if* $x' \neq x$,
- $f(t_1, \ldots, t_n)[x := u] = f(t_1[x := u], \ldots, t_n[x := u])$.

If $(\Theta, x_1 : s_1, \ldots, x_n : s_n, \Psi) \vdash t : s'$ and $\Gamma \vdash u_i : s_i$ for all $1 \leq i \leq n$, let $\Theta, \Gamma, \Psi \vdash t[x_1 := u_1, \ldots, x_n := u_n] : s'$ denote the term obtained by simultaneous substitution, also written $t[x_i := u_i]$ or $t[\vec{x} := \vec{u}]$ for compactness when this is clear.

4.11. DEFINITION. *Let Γ and Θ be contexts over an algebraic signature Σ, where $\Theta = (x_1 : s_1, \ldots, x_n : s_n)$ is finite. A context morphism $\Gamma \to \Theta$ is a tuple of terms $\Gamma \vdash t_i : s_i$ for $1 \leq i \leq n$, written $[x_1 := t_1, \ldots, x_n := t_n] : \Gamma \to \Theta$, or $[x_i := t_i]$ or $[\vec{x} := \vec{t}]$ for compactness.*

If $\Psi = (y_1 : s'_1, \ldots, y_m : s'_m)$ is another finite context, and $[\vec{y} := \vec{u}] : \Theta \to \Psi$ a context morphism, the composition $[\vec{y} := \vec{u}] \circ [\vec{x} := \vec{t}]$ is defined to be $[y_i := u_i[\vec{x} := \vec{t}]] : \Gamma \to \Psi$.

4.12. EXAMPLE. Continuing with Examples 4.4 and 4.9, consider contexts $\Gamma = (x_1 : m,\ x_2 : m,\ p : s)$ and $\Theta = (y : m,\ q : s)$. There is a context morphism $\Gamma \to \Theta$ given by

$$\left[y := x_1, q := \alpha\big(\mu(x_1, x_2), p\big) \right].$$

4.13. DEFINITION. *Let Σ be an algebraic signature. Define the* category of contexts over Σ, *denoted* $\mathbf{Cxt}_\Sigma$, *to be the category of finite contexts over Σ and context morphisms. We define the* category of possibly infinite contexts over Σ, *denoted* $\overline{\mathbf{Cxt}_\Sigma}$, *to be the obvious extension.*

4.14. REMARK. It is not hard to see that $\mathbf{Cxt}_\Sigma$ has finite products, given by concatenation of contexts, and that the objects of $\mathbf{Cxt}_\Sigma$ are freely generated under products by the base sorts (i.e. the singleton contexts). Thus this construction defines a functor $\mathbf{Cxt} \colon \mathbf{ASig} \to \mathbf{ATh}$. In fact, the functor $\mathbf{Cxt}$ is left adjoint to the underlying signature functor $U \colon \mathbf{ATh} \to \mathbf{ASig}$ from Example 4.5. Hence we will also refer to $\mathbf{Cxt}_\Sigma$ as the *free algebraic theory on the signature Σ.*

The category $\mathbf{ASig}$ is for many purposes too rigid: a morphism in $\mathbf{ASig}$ is required to send function symbols to function symbols, whereas one often wants to send function symbols to a more complex expression. We now define this more flexible category of signatures.

4.15. DEFINITION. *Define* $\mathbf{ASig}^*$ *to be the Kleisli category of the monad induced by the adjunction* $\mathbf{Cxt} \dashv U$ *of Remark 4.14 on the category* $\mathbf{ASig}$.[2] *Concretely,* $\mathbf{ASig}^*$ *is defined just like* $\mathbf{ASig}$ *in Definition 4.3, but where a morphism $F \colon \Sigma \to \Sigma'$ between signatures is allowed to send a function symbol $f \colon (s_1, \ldots, s_n) \to s'$ in Φ_Σ to an arbitrary term $\bigl(x_1 : F_S(s_1), \ldots, x_n : F_S(s_n)\bigr) \vdash F_\Phi(f) : F_S(s')$ over Σ'. Composition of these signature morphisms is defined by substitution.*

We are now ready to discuss presentations of algebraic theories. We begin with a careful consideration of equations.

4.16. DEFINITION. *Let Σ be an algebraic signature. An* equation over Σ *is a pair of terms (t, t'), where t and t' are in the same finite context Γ and have the same sort s. We denote such a pair by the equation $\Gamma \vdash (t = t') : s$, or simply by $\Gamma \vdash t = t'$ if no confusion should arise.*

Let E be a set of equations over Σ. Define $\approx_E$ to be the smallest equivalence relation on terms over Σ such that

1. *if $\Gamma \vdash t = t'$ is an equation of E, then $\Gamma \vdash t \approx_E t'$,*
2. *if $f \colon (s_1, \ldots, s_n) \to s'$ is a function symbol and $\Gamma \vdash (t_i \approx_E t_i') : s_i$ for all $1 \leq i \leq n$, then $\Gamma \vdash \bigl(f(t_1, \ldots, t_n) \approx f(t_1', \ldots, t_n')\bigr) : s'$,*
3. *if $\Theta \vdash (t \approx_E t') : s$, and $[\vec{x} := \vec{u}] \colon \Gamma \to \Theta$ is a context morphism, then $\Gamma \vdash \bigl(t[\vec{x} := \vec{u}] \approx_E t'[\vec{x} := \vec{u}]\bigr) : s$.*

4.17. REMARK. Condition 3 of Definition 4.16 is equivalent to the following two conditions:

3a. if $(\Theta, \Psi) \vdash (t \approx_E t') : s'$, then $(\Theta, x : s, \Psi) \vdash (t \approx_E t') : s'$ for any sort s',
3b. if $(\Theta, x : s, \Psi) \vdash (t \approx_E t') : s'$, and $\Gamma \vdash u : s$ is a term, then we have $(\Theta, \Gamma, \Psi) \vdash \bigl(t[x := u] \approx_E t'[x := u]\bigr) : s'$.

---

[2] We use the symbol $\vdash$ between contexts and terms; we use the symbol $\dashv$ for adjunctions. Both are standard notation and no confusion should arise.

4.18. DEFINITION. *Let Σ be an algebraic signature, and E a set of equations over Σ. The algebraic theory $\mathbf{Cxt}_\Sigma/E$ is the quotient of $\mathbf{Cxt}_\Sigma$ by the equivalence relation $\approx_E$. In other words, the objects of $\mathbf{Cxt}_\Sigma/E$ are finite contexts over Σ, and the morphisms are $\approx_E$-equivalence classes of context morphisms. This quotient is well defined because $\approx_E$ is by definition preserved under substitution.*

We call the pair (Σ, E) a presentation *of the algebraic theory $\mathcal{T}$ if there is an isomorphism $\mathcal{T} \cong \mathbf{Cxt}_\Sigma/E$. We call it a* finite presentation *if both Σ and E are finite.*

We now conclude our running example of monoid actions.

4.19. EXAMPLE. In Example 4.4, we gave the signature for monoid actions on sets, with sorts m, s and function symbols η, μ, α. To present the algebraic theory of monoid actions on sets, we add the following four equations:

$$x : m \vdash \mu(x, \eta) = x \qquad\qquad x, y, z : m \vdash \mu\big(x, \mu(y, z)\big) = \mu\big(\mu(x, y), z\big)$$
$$x : m \vdash \mu(\eta, x) = x \qquad x : m,\, y : m,\, p : s \vdash \alpha\big(x, \alpha(y, p)\big) = \alpha\big(\mu(x, y), p\big)$$

4.20. DEFINITION. *Define the category of* algebraic presentations $\mathbf{APr}$ *as follows: the objects of $\mathbf{APr}$ are pairs (Σ, E), where Σ is an algebraic signature and E is a set of equations over Σ. A morphism $F \colon (\Sigma, E) \to (\Sigma', E')$ is a morphism $F \colon \Sigma \to \Sigma'$ in the Kleisli category $\mathbf{ASig}^*$ such that $F(t) \approx_{E'} F(t')$ for each equation $t = t'$ of E.*

Let $\mathbf{Cxt}$ also denote the functor $\mathbf{APr} \to \mathbf{ATh}$ sending a pair (Σ, E) to $\mathbf{Cxt}_\Sigma/E$.

4.21. REMARK. Any algebraic theory $\mathcal{T}$ has a *canonical presentation* $(\Sigma_\mathcal{T}, E_\mathcal{T})$, where $\Sigma_\mathcal{T}$ is the underlying signature from Example 4.5, and $E_\mathcal{T}$ is defined such that an equation $x_1 : s_1, \ldots, x_n : s_n \vdash (t = t') : s$ is in $E_\mathcal{T}$ if and only if the morphisms corresponding to t and t' in the hom-set $\mathcal{T}(s_1 \times \cdots \times s_n, s)$ are equal.

It is not hard to see that $\mathbf{Cxt}_{\Sigma_\mathcal{T}}/E_\mathcal{T} \cong \mathcal{T}$ for any algebraic theory $\mathcal{T}$. It is also straightforward to check that $\mathbf{Cxt} \colon \mathbf{APr} \to \mathbf{ATh}$ is fully faithful, and hence an equivalence of categories.

The following easy proposition establishes the fundamental connection between a presentation for an algebraic theory $\mathcal{T}$ and algebras on $\mathcal{T}$.

4.22. PROPOSITION. *Let Σ be an algebraic signature and E be a set of equations, and consider an assignment of a set F_s to each sort $s \in S_\Sigma$ and a function $F_f \colon F_{s_1} \times \cdots \times F_{s_n} \to F_{s'}$ to each function symbol $f \colon (s_1, \ldots, s_n) \to s'$ in Φ_Σ. This assignment uniquely extends to a $\mathbf{Cxt}_\Sigma$-algebra F. In particular, given any term $(x_1 : s_1, \ldots, x_n : s_n) \vdash t : s'$, there is a function $F_t \colon F_{s_1} \times \cdots \times F_{s_n} \to F_{s'}$.*

The assignment uniquely extends to a $\mathbf{Cxt}_\Sigma/E$-algebra if and only if it satisfies the equations E, i.e. for each equation $\Gamma \vdash t_1 = t_2$ of E, the functions F_{t_1} and F_{t_2} are equal.

4.23. EXAMPLE. Consider the presentation (Σ, E), where $S_\Sigma = \{\mathsf{Int}\}$ is the only sort, Φ_Σ consists of the five function symbols $0, 1 \colon () \to \mathsf{Int}$, $(-) \colon (\mathsf{Int}) \to \mathsf{Int}$, and $+, \times \colon (\mathsf{Int}, \mathsf{Int}) \to \mathsf{Int}$, and E is the set of equations shown in Fig. 1. The algebraic theory $\mathcal{T} = \mathbf{Cxt}_\Sigma/E$ generated by this presentation is a category with objects the contexts over Σ, such as

$$x, y, z : \mathsf{Int} \vdash (x + y) + z = x + (y + z)$$
$$x : \mathsf{Int} \vdash x + 0 = x$$
$$x, y : \mathsf{Int} \vdash x + y = y + x$$
$$x : \mathsf{Int} \vdash x + (-x) = 0$$
$$x, y, z : \mathsf{Int} \vdash (x \times y) \times z = x \times (y \times z)$$
$$x : \mathsf{Int} \vdash x \times 1 = x$$
$$x, y : \mathsf{Int} \vdash x \times y = y \times x$$
$$x, y, z : \mathsf{Int} \vdash x \times (y + z) = (x \times y) + (x \times z)$$

Figure 1: Equations for the algebraic theory of commutative rings.

$(a : \mathsf{Int})$ or $(x, y, z : \mathsf{Int})$. Some example context morphisms $(x, y, z : \mathsf{Int}) \to (a : \mathsf{Int})$ are $[a := (x + y) \times (x + z)]$ and $[a := ((x \times x) + (x \times z)) + ((y \times x) + (y \times z))]$. These context morphisms are equivalent under $\approx_E$, so determine the same morphism in $\mathcal{T}$.

It is possible to show that the category $\mathcal{T}$-**Alg** is equivalent to the category **CRing** of commutative rings. In particular, a product preserving functor $F : \mathcal{T} \to \mathbf{Set}$ must send the context $(a : \mathsf{Int})$ to some set R, the context morphism $[a := x + y] : (x, y : \mathsf{Int}) \to (a : \mathsf{Int})$ to a function $R \times R \to R$, etc.

We name the single sort 'Int' to fit with the practice in type theory, in which the "elements" of a type τ are considered to be the ground terms of type τ. That the ground terms of sort 'Int' are precisely the integers is equivalent to the fact that $\mathbb{Z}$ is the initial commutative ring.

We will use the following algebraic theory throughout the paper in our database examples.

4.24. EXAMPLE. Consider the multi-sorted algebraic theory **Type** generated by the finite theory presentation with base sorts, function symbols and equations as defined in Fig. 2, page 573. It may be helpful to recall that implication can be written as $(a \Rightarrow b) = \neg a \vee b$. We use an axiomatization of Boolean algebras which is proven complete in [Hun04].

Clearly this algebraic theory includes the one from Example 4.23 as a sub-theory. Similarly to viewing ground terms of type 'Int' as the integers, those of type 'Str' are strings of letters, presented as the free monoid on 52 generators (upper and lower case letters). For example, when we later write 'Admin$'$: Str we actually mean the term $\varnothing \vdash$ 'A$'$.'d$'$.'m$'$.'i$'$.'n$'$: Str.[3] It can be shown that the ground terms of type 'Bool' are $\{\text{True}, \text{False}\}$.

4.25. PRESENTATIONS OF ALGEBRAS. We now turn to presentations of algebras. Fix a presentation (Σ, E), and let $\mathcal{T} = \mathbf{Cxt}_\Sigma / E$ be the presented algebraic theory. Recall by Definition 4.6 how we can think of objects in $\mathbf{Set}_{/S_\Sigma}$, the category of S_Σ-indexed sets, as (possibly infinite) contexts over Σ. There is an evident forgetful functor $U : \mathcal{T}$-$\mathbf{Alg} \to \mathbf{Set}_{/S_\Sigma}$, which sends an algebra $A : \mathcal{T} \to \mathbf{Set}$ to the indexed set $\{(UA)_s\}_{s \in S_\Sigma}$ where $(UA)_s = A(s)$.

---

[3] Similarly, we may use the shorthand $x - y$ to denote what is really $x + (-y)$.

S_Σ : Int, Bool, Str

Φ_Σ :

$0, 1 : () \to$ Int	$\top, \bot : () \to$ Bool	$\varepsilon,$ 'a'$, \ldots,$ 'Z'$: () \to$ Str
$(-) : ($Int$) \to$ Int	$\neg : ($Bool$) \to$ Bool	$(.) : ($Str$,$ Str$) \to$ Str
$+, \times : ($Int$,$ Int$) \to$ Int	$\wedge : ($Bool$,$ Bool$) \to$ Bool	eq $: ($Str$,$ Str$) \to$ Bool
$\leq : ($Int$,$ Int$) \to$ Bool	$\vee : ($Bool$,$ Bool$) \to$ Bool	

E : *boolean algebra:*

$$\alpha : \mathsf{Bool} \vdash \alpha \vee \bot = \alpha \qquad\qquad \alpha : \mathsf{Bool} \vdash \alpha \wedge \top = \alpha$$
$$\alpha, \beta : \mathsf{Bool} \vdash \alpha \vee \beta = \beta \vee \alpha \qquad\qquad \alpha, \beta : \mathsf{Bool} \vdash \alpha \wedge \beta = \beta \wedge \alpha$$
$$\alpha : \mathsf{Bool} \vdash \alpha \vee \neg\alpha = \top \qquad\qquad \alpha : \mathsf{Bool} \vdash \alpha \wedge \neg\alpha = \bot$$
$$\alpha, \beta, \gamma : \mathsf{Bool} \vdash \alpha \vee (\beta \wedge \gamma) = (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \quad \alpha, \beta, \gamma : \mathsf{Bool} \vdash \alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

commutative ring: all equations from Fig. 1

totally pre-ordered ring:

$$x, y, z : \mathsf{Int} \vdash \neg((x \leq y) \wedge (y \leq z)) \vee (x \leq z) = \top$$
$$x, y : \mathsf{Int} \vdash (x \leq y) \vee (y \leq x) = \top$$
$$x, y, z, w : \mathsf{Int} \vdash \neg((x \leq y) \wedge (z \leq w)) \vee (x + z \leq y + w) = \top$$
$$x, y, z : \mathsf{Int} \vdash \neg((x \leq y) \wedge (0 \leq z)) \vee (x \times z \leq y \times z) = \top$$
$$x, y, z : \mathsf{Int} \vdash \neg((x \times z \leq y \times z) \wedge (0 \leq z)) \vee (x \leq y) = \top$$
$$\varnothing \vdash (1 \leq 0) = \neg\top$$

monoid:

$$s : \mathsf{Str} \vdash s.\varepsilon = s \quad s, t, u : \mathsf{Str} \vdash (s.t).u = s.(t.u)$$
$$s : \mathsf{Str} \vdash \varepsilon.s = s$$

congruence:

$$s : \mathsf{Str} \vdash (s \text{ eq } s) = \top$$
$$s, t : \mathsf{Str} \vdash (s \text{ eq } t) = (t \text{ eq } s)$$
$$s, t, u : \mathsf{Str} \vdash \neg((s \text{ eq } t) \wedge (t \text{ eq } u)) \vee (s \text{ eq } u) = \top$$
$$s, t, u, v : \mathsf{Str} \vdash \neg((s \text{ eq } t) \wedge (u \text{ eq } v)) \vee (s.u \text{ eq } t.v) = \top$$

decidable equality:

$$s, t, u : \mathsf{Str} \vdash (s.u \text{ eq } t.u) = (s \text{ eq } t)$$
$$s, t, u : \mathsf{Str} \vdash (s.t \text{ eq } s.u) = (t \text{ eq } u)$$

$s, t : \mathsf{Str} \vdash (s.\text{'a'} \text{ eq } t.\text{'b'}) = \neg\top$	$\ldots$	$s, t : \mathsf{Str} \vdash (s.\text{'y'} \text{ eq } t.\text{'z'}) = \neg\top$
$s, t : \mathsf{Str} \vdash (\text{'a'}.s \text{ eq } \text{'b'}.t) = \neg\top$	$\ldots$	$s, t : \mathsf{Str} \vdash (\text{'y'}.s \text{ eq } \text{'z'}.t) = \neg\top$
$s : \mathsf{Str} \vdash (s.\text{'a'} \text{ eq } \epsilon) = \neg\top$	$\ldots$	$s : \mathsf{Str} \vdash (s.\text{'z'} \text{ eq } \epsilon) = \neg\top$
$s : \mathsf{Str} \vdash (\text{'a'}.s \text{ eq } \epsilon) = \neg\top$	$\ldots$	$s : \mathsf{Str} \vdash (\text{'z'}.s \text{ eq } \epsilon) = \neg\top$

Figure 2: Presentation of **Type**, our running example of an algebraic theory.

4.26. DEFINITION. *Let $\Gamma \in \mathbf{Set}_{S_\Sigma}$ be an S_Σ-indexed set, thought of as a context, and let $\mathcal{T} = \mathbf{Cxt}_\Sigma/E$ be a presented algebraic theory. Define the* free $\mathcal{T}$-algebra on Γ, *denoted $\kappa[\Gamma]\colon \mathcal{T} \to \mathbf{Set}$, to be the algebra for which $\kappa[\Gamma](\Theta)$ is the set of $\approx_E$-equivalence classes of context morphisms $\Gamma \to \Theta$, with functoriality given by composition of context morphisms.*

4.27. REMARK. With notation as in Definition 4.26, the elements of $\kappa[\Gamma]$ of sort s are just the $\approx_E$-equivalence classes of terms $\Gamma \vdash t : s$, and for any function symbol $f \colon (s_1, \ldots, s_n) \to s'$ the induced function sends a tuple of terms $\Gamma \vdash t_i : s_i$ to the term $\Gamma \vdash f(t_1, \ldots, t_n) : s'$. By Proposition 4.22, this completely defines the algebra $\kappa[\Gamma]$, whose standard name is the *term algebra* over Γ. There is an adjunction

$$\kappa[-]\colon \mathbf{Set}_{S_\Sigma} \leftrightarrows (\mathbf{Cxt}_\Sigma/E)\text{-}\mathbf{Alg} : U \tag{22}$$

For this reason, we may refer to $\kappa[\Gamma]$ as the *free algebra* on the generating context Γ.

4.28. EXAMPLE. Let $\mathcal{T}$ be any algebraic theory. The algebra generated by the empty context is the algebra of constants $\kappa[\emptyset] = \kappa = \mathbf{y}(\mathbf{1})$; see Example 3.4. Note that any term in κ is necessarily a ground term (Definition 4.8). If X is any other $\mathcal{T}$-algebra, we refer to terms in the image of the unique map $\kappa \to X$ as *constants* in X.

4.29. EXAMPLE. Let $\mathcal{T}$ be the algebraic theory from Example 4.23. The elements of $\kappa[x, y : \mathsf{Int}]$ of the unique base sort 'Int' are $\approx_E$-equivalence classes of terms $x, y : \mathsf{Int} \vdash t : \mathsf{Int}$, such as $x, y : \mathsf{Int} \vdash (x + y) \times x$. But these are just polynomials in the variables x and y, hence the commutative ring $\kappa[x, y : \mathsf{Int}] \in \mathcal{T}\text{-}\mathbf{Alg}$ is the polynomial ring $\mathbb{Z}[x, y]$, the free commutative ring on the set $\{x, y\}$.

4.30. REMARK. Note that the Kleisli category for the adjunction (22) is precisely the opposite of the category $\overline{\mathbf{Cxt}_\Sigma}/E$ of possibly infinite contexts (Definition 4.13), and the restriction of this Kleisli category to those objects $X \to S_\Sigma$ of $\mathbf{Set}_{/S_\Sigma}$ for which X is finite is the category $(\mathbf{Cxt}_\Sigma/E)^{\mathrm{op}}$. Another way to say this is that the algebraic theory $\mathbf{Cxt}_\Sigma/E$ is isomorphic to the opposite of the category of finitely generated free algebras over $\mathbf{Cxt}_\Sigma/E$, a fact which is true for any algebraic theory; see [ARV11, § 8].

4.31. DEFINITION. *Let Σ be an algebraic signature, Γ a context over Σ, and e an equation over Σ. Say that e is an* equation in Γ *if it is between terms in context Γ. A set E' of equations over Σ is said to be* in Γ *if each element $e \in E'$ is.*

4.32. DEFINITION. *Let (Σ, E) be a presentation for an algebraic theory $\mathcal{T}$. A $\mathcal{T}$-algebra presentation is a pair (Γ, E'), where Γ is a context over Σ, and E' is a set of equations in Γ. Define $\kappa[\Gamma]/E'$ to be the quotient of the free $\mathcal{T}$-algebra $\kappa[\Gamma]$ (Definition 4.26) by the equations E'. Concretely, $(\kappa[\Gamma]/E')(\Theta)$ is the set of $\approx_{E \cup E'}$-equivalence classes of context morphisms $\Gamma \to \Theta$.*

A morphism of $\mathcal{T}$-algebra presentations (cf. Definition 4.20) $(\Gamma', E') \to (\Gamma'', E'')$ is simply a context morphism $[\vec{x} \coloneqq \vec{t}]\colon \Gamma'' \to \Gamma'$ (note the direction!) such that for each equation $\Gamma' \vdash u = v$ in E', it follows that $\Gamma'' \vdash u[\vec{x} \coloneqq \vec{t}] \approx_{E \cup E''} v[\vec{x} \coloneqq \vec{t}]$.

4.33. EXAMPLE. Let (Σ, E) be the theory of commutative rings as in Example 4.23, and let $\Gamma = (x, y : \mathsf{Int})$. Then $\kappa[\Gamma]$ is the polynomial ring $\mathbb{Z}[x, y]$ (Example 4.29). If e is the equation $x^3 = y^2$ then $(\kappa[\Gamma]/\{e\})$ is the ring $\mathbb{Z}[x, y]/(x^3 - y^2)$.

4.34. REMARK. Recall that by Remark 4.21, every algebraic theory has a canonical presentation and the functor $\mathbf{APr} \to \mathbf{ATh}$ from presentations to theories is an equivalence. For algebras the same turns out to be true. First, every $\mathcal{T}$-algebra $A \in (\mathbf{Cxt}_\Sigma/E)\text{-}\mathbf{Alg}$ has a canonical presentation (Γ, E'), where Γ is the underlying S_Σ-indexed set UA, and E' is the set of equations $\Gamma \vdash t = t'$ for which t and t' are equated under the counit $\kappa[\Gamma] \to A$ of the adjunction from (22). Second, the category of such presentations (whose objects and morphisms are given in Definition 4.32) is equivalent to $\mathcal{T}\text{-}\mathbf{Alg}$.

4.35. PRESENTATIONS OF CATEGORIES. It is well known that categories are algebraic over directed graphs, i.e. that a category can be presented by giving a graph together with a set of equations (see e.g. [Mac98, § II.8]). In the interest of completeness and consistency, we will show here how to consider presentations for categories as a special case of presentations for algebraic theories (see Definition 4.41).

Formally, a directed graph G consists of a set G_0 of nodes and a set G_1 of edges, together with functions $\mathrm{dom}, \mathrm{cod} \colon G_1 \to G_0$. Note that a directed graph G can be seen as an algebraic signature (Definition 4.2) in which all function symbols are unary. The set of sorts of the unary signature is simply the set G_0 of nodes of G, and the set G_1 of edges is taken as the set of function symbols.

4.36. DEFINITION. *Say that an algebraic signature Σ is* unary *when all of its function symbols are unary. As usual, we will write $f \colon A \to B$ as shorthand for $\mathrm{dom}(f) = A$ and $\mathrm{cod}(f) = B$. From now on, we will identify a graph G with its corresponding unary algebraic signature.*

4.37. REMARK. Let G be a graph, and let $A, B \in G_0$ be nodes. Terms (Definition 4.8) of type B in a singleton context $(x : A)$ over (the unary signature associated to) G can be identified with paths from A to B in G. As G has only unary function symbols, all such terms must be of the form $x : A \vdash f_n(\ldots f_2(f_1(x))) : B$ for some $n \geq 0$.

4.38. PROPOSITION. *Let G be a graph and $\mathrm{Fr}(G)$ the free category generated by G. Then $\mathrm{Fr}(G)$ is isomorphic to the full subcategory of $\mathbf{Cxt}_G$ spanned by the singleton contexts.*

4.39. NOTATION. Let Σ be an algebraic signature, and let $\Gamma \vdash t$ be a term in some context. In order to reduce parentheses, we will use the notation $\Gamma \vdash t.f_1.f_2 \ldots f_n$ to denote $\Gamma \vdash f_n(\ldots f_2(f_1(t)))$, assuming that this is a well-formed term and that each f_i is unary.

4.40. DEFINITION. *Let Σ be a (not necessarily unary) signature. An equation $\Gamma \vdash t = t'$ over Σ is* unary *if the context Γ is a singleton. Say that a set E of equations is* unary *if it consists only of unary equations.*

4.41. DEFINITION. *A* category presentation *is a pair* (G, E), *where G is a graph and E is a set of unary equations over G. Define the category* presented *by* (G, E), *denoted* $\mathrm{Fr}(G)/E$, *to be the full subcategory of* $\mathbf{Cxt}_G/E$ *spanned by the singleton contexts.*

4.42. PROPOSITION. *Let* (G, E) *be a category presentation. The category* $\mathbf{Cxt}_G/E$ *is the free category-with-finite-products on the category* $\mathrm{Fr}(G)/E$. *In particular, there is an equivalence of categories* $(\mathbf{Cxt}_G/E)\text{-}\mathbf{Alg} \simeq [\mathrm{Fr}(G)/E, \mathbf{Set}]$.

4.43. PRESENTATIONS OF SET-VALUED FUNCTORS. If $\mathscr{C}$ is a category given by a presentation (G, E), then Proposition 4.42 provides a way of giving presentations for functors $\mathscr{C} \to \mathbf{Set}$. Let Γ be a context over the unary algebraic signature G. Then we can form the free algebra $\kappa[\Gamma] \in (\mathbf{Cxt}_G/E)\text{-}\mathbf{Alg}$ as in Definition 4.26. Under the equivalence $(\mathbf{Cxt}_G/E)\text{-}\mathbf{Alg} \simeq [\mathscr{C}, \mathbf{Set}]$, this corresponds to a functor $\mathscr{C} \to \mathbf{Set}$, namely the restriction of $\kappa[\Gamma]\colon \mathbf{Cxt}_G/E \to \mathbf{Set}$ to its full subcategory of singleton contexts $\mathscr{C}$. We will denote this restriction $\langle\Gamma\rangle$.

It is straightforward to check that the adjunction from Remark 4.27 restricts to an adjunction $\langle-\rangle\colon \mathbf{Set}_{G_0} \leftrightarrows [\mathscr{C}, \mathbf{Set}] \colon U$. Hence $\langle\Gamma\rangle$ is the *free copresheaf* on $\mathscr{C}$ generated by Γ.

Similarly, if E' is a set of equations in context Γ, as in Definition 4.31, then we denote by $\langle\Gamma\rangle/E'$ the restriction of $\kappa[\Gamma]/E'\colon \mathbf{Cxt}_G/E \to \mathbf{Set}$ to $\mathscr{C}$, and refer to this as the copresheaf *presented* by (Γ, E').

4.44. PRESENTATIONS OF ALGEBRAIC PROFUNCTORS. In Section 5 we will be interested in algebraic profunctors $M\colon \mathscr{C} \nrightarrow \mathscr{T}$ where $\mathscr{C}$ is a category and $\mathscr{T}$ is an algebraic theory; see Definition 3.10. Our approach to presenting an algebraic profunctor M between $\mathscr{C} = \mathrm{Fr}(G)/E_G$ and $\mathscr{T} = \mathbf{Cxt}_\Sigma/E_\Sigma$ will be in terms of its collage $\widetilde{M}$, as in Example 2.19.

4.45. DEFINITION. *Let* $G = (G_0, G_1)$ *be a graph (unary signature) and* $\Sigma = (S_\Sigma, \Phi_\Sigma)$ *be an algebraic signature. A* profunctor signature Υ *from G to Σ is a set of unary function symbols, where each function symbol* $\mathtt{att} \in \Upsilon$ *is assigned a sort* $a := \mathrm{dom}(\mathtt{att}) \in G_0$ *and a sort* $\tau := \mathrm{cod}(\mathtt{att}) \in S_\Sigma$. *We will sometimes refer to the function symbol* $\mathtt{att} \in \Upsilon$ *as an* attribute, *and denote it* $\mathtt{att}\colon a \to \tau$.

A profunctor signature Υ *has an associated algebraic signature* $\widetilde{\Upsilon} = (S_{\widetilde{\Upsilon}}, \Phi_{\widetilde{\Upsilon}})$, *with sorts* $S_{\widetilde{\Upsilon}} = G_0 \sqcup S_\Sigma$, *and function symbols* $\Phi_{\widetilde{\Upsilon}} = G_1 \sqcup \Upsilon \sqcup \Phi_\Sigma$.

Say that a set E_Υ *of equations over* $\widetilde{\Upsilon}$ *is a set of* profunctor equations *if for each equation* $\Gamma \vdash (t_1 = t_1) : s'$ *in* E_Υ, *the context is a singleton* $\Gamma = (x : s)$ *with* $s \in G_0$ *and* $s' \in S_\Sigma$.

4.46. DEFINITION. *Let* (G, E_G) *be a category presentation,* (Σ, E_Σ) *an algebraic theory presentation,* Υ *a profunctor signature from G to Σ, and* E_Υ *a set of profunctor equations. Let* $E_{\widetilde{\Upsilon}} = E_G \cup E_\Sigma \cup E_\Upsilon$. *Define the algebraic profunctor* presented *by this data, denoted* $\kappa[\Upsilon]/E_\Upsilon\colon \mathrm{Fr}(G)/E_G \nrightarrow \mathbf{Cxt}_\Sigma/E_\Sigma$, *as follows:*
  - *for any node* $a \in G_0$ *and context* $\Gamma \in \mathbf{Cxt}_\Sigma$, *the set* $(\kappa[\Upsilon]/E_\Upsilon)(a, \Gamma)$ *is the hom set* $(\mathbf{Cxt}_{\widetilde{\Upsilon}}/E_{\widetilde{\Upsilon}})((x : a), \Gamma)$, *i.e. the set of* $\approx_{E_G \cup E_\Sigma \cup E_\Upsilon}$*-equivalence classes of context morphisms* $(x : a) \to \Gamma$ *over* $\widetilde{\Upsilon}$,

- *the functorial actions are given by substitution.*

It is clear from the definition that the collage of the profunctor $\kappa[\Upsilon]/E_\Upsilon$ is a full subcategory of $\mathbf{Cxt}_{\widetilde{\Upsilon}}/E_{\widetilde{\Upsilon}}$. In fact, it is not much harder to see the following proposition; cf. Proposition 4.42.

4.47. PROPOSITION. *Let $\mathscr{C}$ be a category with presentation (G, E_G), let $\mathscr{T}$ be an algebraic theory with presentation (Σ, E_Σ), and let $P \colon \mathscr{C} \nrightarrow \mathscr{T}$ be an algebraic profunctor with presentation (Υ, E_Υ). The category $\mathbf{Cxt}_{\widetilde{\Upsilon}}/E_{\widetilde{\Upsilon}}$ is the free completion of the collage $\widetilde{P}$ under finite products for which existing products in $\mathscr{T}$ are preserved. In particular, the category $(\mathbf{Cxt}_{\widetilde{\Upsilon}}/E_{\widetilde{\Upsilon}})$-$\mathbf{Alg}$ of functors $(\mathbf{Cxt}_{\widetilde{\Upsilon}}/E_{\widetilde{\Upsilon}}) \to \mathbf{Set}$ which preserve all finite products is equivalent to the category of functors $\widetilde{P} \to \mathbf{Set}$ whose restriction to $\mathscr{T}$ preserves finite products.*

4.48. EXAMPLE. Let $\mathscr{C}$ be the category presented by the terminal graph $G_0 = \{X\}$, $G_1 = \{f\}$, with equation $x : X \vdash x.f = x.f.f$. Let $\mathscr{T}$ be the algebraic theory of commutative rings, as in Example 4.23. Consider the algebraic profunctor $M \colon \mathscr{C} \nrightarrow \mathscr{T}$ presented by a single attribute $\Upsilon = \{p \colon X \to \mathsf{Int}\}$ and a single equation $E = \{x : X \vdash x.f.p = x.p \times x.p\}$. One can check that this presents the following profunctor $\mathscr{C} \nrightarrow \mathscr{T}$:

$$\begin{aligned} \kappa[\Upsilon]/E_\Upsilon &\cong \mathbb{Z}[x.f^n.p]/\big(\ x.f^{n+1}.p = (x.f^n.p)^2\ ,\ \ x.f^{n+2}.p = x.f^{n+1}.p\ \big) \\ &\cong \mathbb{Z}[y_0, y_1]/\big(\ y_1 = y_0^2\ ,\ \ y_1 = y_1^2\ \big) \\ &\cong \mathbb{Z}[y]/\big(\ y^2 = y^4\ \big) \end{aligned}$$

where, in the first line, n ranges over all natural numbers. The edge $f \in G_1$ induces the ring endomorphism $f(y) \mapsto y^2$.

5. Algebraic database schemas

In this section we move beyond background and into our construction of databases. What we call (algebraic) databases straddle what are traditionally known as relational databases and the more modern graph databases. Importantly, algebraic databases also integrate a programming language **Type**, by which to operate on attribute values.

We take our terminology from the relational database world. That is, a database consists of a conceptual layout, called a *schema*, as well as some conforming data, called an *instance* (because it represents our knowledge in the current instant of time). In this section we discuss the category of schemas; in Section 6 we discuss instances on them.

5.1. SCHEMAS. For the rest of the paper, **Type** will be an arbitrary multi-sorted finitely presented algebraic theory, as defined in Definition 4.18. However, in all examples, we will fix **Type** to be the algebraic theory described in Example 4.24. Recall from Definition 3.10 the notion of algebraic profunctors $M \in \mathbf{Prof}^\times(\mathscr{C}, \mathbf{Type})$, denoted $M \colon \mathscr{C} \nrightarrow \mathbf{Type}$.

5.2. DEFINITION. *A database schema* **S** *over* **Type** *is a pair* (S_e, S_o), *where*
- S_e *is a category, and*
- $S_o \colon S_e \nrightarrow \textbf{Type}$ *is an algebraic profunctor; i.e.* $S_o \in \textbf{Prof}^\times(S_e, \textbf{Type})$.

We refer to S_e *as the* entity category *of* **S** *and to* S_o *as the* observables *profunctor. We will also write* $S_o \colon S_e^{\mathrm{op}} \to \textbf{Type-Alg}$ *for the exponential transpose of* $S_o \colon S_e^{\mathrm{op}} \times \textbf{Type} \to \textbf{Set};$ *see Lemma 3.9.*

5.3. REMARK. It is often convenient to work with schemas in terms of their collages. If **S** is the schema $S_o \colon S_e \nrightarrow \textbf{Type}$, we write $\widetilde{S}$ for the collage of the profunctor S_o; see Example 2.19. By (13), it comes equipped with a map $\widetilde{S} \to \textbf{2}$ and we refer to the two pullbacks below respectively as the *entity side* and the *type side* of the collage:

$$
\begin{array}{ccccc}
S_e & \xrightarrow{\;i_S\;} & \widetilde{S} & \xleftarrow{\;i_\textbf{T}\;} & \textbf{Type} \\
\scriptstyle! \downarrow & \llcorner & \downarrow & \lrcorner & \downarrow \scriptstyle! \\
\{*\} & \xrightarrow{\;0\;} & \textbf{2} & \xleftarrow{\;1\;} & \{*\}
\end{array}
$$

5.4. EXAMPLE. Any **Type**-algebra $X \colon \textbf{Type} \to \textbf{Set}$ can be regarded as a schema $(\{*\}, X)$, where the entity category is terminal. In particular, the initial **Type**-algebra κ, described in Example 3.4, can be viewed as a schema $\textbf{U} = (\{*\}, \kappa)$ called the *unit schema*.[4]

5.5. PRESENTATIONS OF SCHEMAS. A presentation for a schema $S_o \colon S_e \nrightarrow \textbf{Type}$ is simply a presentation for the category S_e (see Definition 4.41) together with a presentation for the algebraic profunctor S_o (see Definition 4.46). We spell this out in Definition 5.6.

5.6. DEFINITION. *A* schema signature $\Xi = (G_\Xi, \Upsilon_\Xi)$ *consists of a graph* G_Ξ *together with a profunctor signature* Υ_Ξ *from* G_Ξ *to the signature of* **Type**.

A schema presentation (Ξ, E_Ξ) *consists of a schema signature* Ξ, *together with equations* $E_\Xi = (E_e, E_o)$, *where* E_e *is a set of unary equations over* G_Ξ, *and* E_o *is a set of profunctor equations over* Υ_Ξ. *Note that* (G_Ξ, E_e) *is a presentation for a category, which will be the entity category* S_e, *and* (Υ_Ξ, E_o) *is a presentation for an algebraic profunctor* $S_e \nrightarrow \textbf{Type}$. *We denote the presented schema by* $\mathrm{Fr}(\Xi)/E_\Xi$.

We will write $\widetilde{\Xi}$ *to mean the associated algebraic signature* $\widetilde{\Upsilon}_\Xi$ *as in Definition 4.45, with sorts* $(G_\Xi)_0 \sqcup S_\Sigma$ *and function symbols* $(G_\Xi)_1 \sqcup \Upsilon_\Xi \sqcup \Phi_\Sigma$, *where* $\textbf{Type} \cong \textbf{Cxt}_\Sigma/E_\Sigma$.

In what follows, we refer to function symbols in Υ_Ξ as *attributes*, and refer to a general term $(x : A) \vdash t : \tau$, where $A \in (G_\Xi)_0$ and $\tau \in \textbf{Type}$, as an *observable on* A *of type* τ. In other words, for a schema $S_o \colon S_e \nrightarrow \textbf{Type}$ and objects $A \in S_e$ and $\tau \in \textbf{Type}$, an observable on A of type τ is an element $t \in S_o(A, \tau)$.

5.7. EXAMPLE. The unit $\textbf{U} = (\{*\}, \kappa)$ of Example 5.4 is presented by the graph with one vertex and no edges, the empty profunctor signature, and no equations. That is, **U** has no attributes, so each of its observables is a ground term $\varnothing \vdash c : \tau$, i.e. a constant $c \in \kappa(\tau) = \textbf{Type}(1, \tau)$.

---

[4]**U** is the unit of a certain symmetric monoidal structure on **Schema** (Definition 5.11), whose restriction to entities is the cartesian monoidal structure on **Cat**; however, we do not pursue that here.

Entities:	Emp, Dept		Attributes:	last : Emp $\to$ Str
Edges:	mgr : Emp $\to$ Emp			name : Dept $\to$ Str
	wrk : Emp $\to$ Dept			sal : Emp $\to$ Int
	sec : Dept $\to$ Emp		Obs. Eqs:	e : Emp $\vdash$
Path Eqs:	e : Emp $\vdash e.\mathtt{mgr}.\mathtt{mgr} = e.\mathtt{mgr}$			$(e.\mathtt{sal} \leq e.\mathtt{mgr}.\mathtt{sal}) = \top$
	e : Emp $\vdash e.\mathtt{mgr}.\mathtt{wrk} = e.\mathtt{wrk}$			
	d : Dept $\vdash d.\mathtt{sec}.\mathtt{wrk} = d$			

Figure 3: Presentation of **S**, our running example of a schema.

5.8. EXAMPLE. Let **Type** be as in Example 4.24. Consider the presentation $(\Xi, E_{\mathrm{e}}, E_{\mathrm{o}})$ for a schema **S** as displayed in Fig. 3, which will serve as a motivating example throughout the paper. In this presentation, we use the labels "Entities" for $(G_\Xi)_0$, "Edges" for $(G_\Xi)_1$, "Attributes" for Υ_Ξ, "Path Eqs" for E_{e}, and "Obs. Eqs" for E_{o}.

Below (23) is a graphical display of this presentation; its two grey dots are the entities, its six arrows are the edges and attributes, and its four equations are the path and observable equations.

$$(23)$$
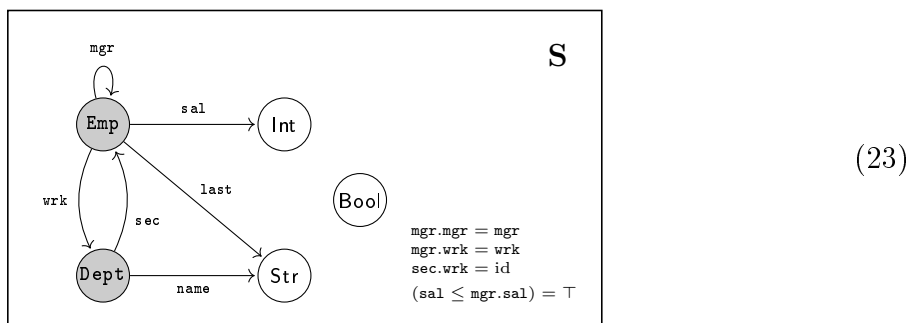
The presented schema **S** is built according to Definitions 4.41 and 4.46, as we now describe explicitly. The entity category S_{e} is the free category on the subgraph of grey objects and arrows between them, modulo the top three equations. An example (context) morphism Emp $\to$ Dept in S_{e} is given by the path mgr.wrk.sec.mgr.wrk. From the equations, we can show that it is equivalent to wrk,

$$e : \mathtt{Emp} \vdash (e.\mathtt{mgr}.\mathtt{wrk}.\mathtt{sec}.\mathtt{mgr}.\mathtt{wrk} \approx e.\mathtt{wrk}) : \mathtt{Dept}$$

In other words, these two terms name the same morphism in S_{e}.

The observables profunctor $S_{\mathrm{o}} : S_{\mathrm{e}} \nrightarrow \mathbf{Type}$ is freely generated by the three arrows from an S_{e}-object $\bullet$ to a **Type**-object $\circ$, modulo the fourth equation. An example observable Dept $\to$ Bool, i.e. an element of $S_{\mathrm{o}}(\mathsf{Dept}, \mathsf{Bool})$, is "whether a department d is named Admin", given by the term $(d : \mathtt{Dept}) \vdash \mathrm{eq}(d.\mathtt{name}, \mathrm{Admin})$. By the fourth equation, we can show it is equivalent to a more complex observable,

$$d : \mathtt{Dept} \vdash ((d.\mathtt{sec}.\mathtt{sal} \leq d.\mathtt{sec}.\mathtt{mgr}.\mathtt{sal}) \wedge \mathrm{eq}(d.\mathtt{name}, \mathrm{Admin})) : \mathsf{Bool}.$$

The schema $\mathbf{S}$ can accommodate database instances in some company setting, as we will see in Example 6.3. In such, there exist tables of employees and departments. In each there are columns (sometimes called *foreign keys*) that reference other tables in order to state where an employee works, who is the departmental secretary, etc. There are also columns that state the last name and salary of each employee, etc. The equations express integrity constraints, e.g. the fact that the secretary of a department works therein, or that every employee is paid less than his or her manager.

5.9. SCHEMA MAPPINGS. We now discuss morphisms of schemas, also known as schema mappings [DHI12]. These will eventually be the vertical morphisms in a proarrow equipment. Recall by Definition 3.10 that a morphism between two algebraic profunctors is just a 2-cell between profunctors as in (9).

5.10. DEFINITION. *A schema mapping* $\mathbf{F}\colon \mathbf{S} \to \mathbf{T}$ *is a pair* $(F_{\mathrm{e}}, F_{\mathrm{o}})$, *where*
- $F_{\mathrm{e}}\colon S_{\mathrm{e}} \to T_{\mathrm{e}}$ *is a functor, and*
- F_{o} *is a 2-cell in* $\mathbb{P}\mathbf{rof}$

$$
\begin{array}{ccc}
S_{\mathrm{e}} & \xrightarrow{\ S_{\mathrm{o}}\ } & \mathbf{Type} \\
{\scriptstyle F_{\mathrm{e}}}\big\downarrow & \Downarrow F_{\mathrm{o}} & \big\| \\
T_{\mathrm{e}} & \xrightarrow[\ T_{\mathrm{o}}\ ]{} & \mathbf{Type}
\end{array}
$$

We will also write F_{o} *for the corresponding natural transformation* $F_{\mathrm{o}}\colon S_{\mathrm{o}} \Rightarrow T_{\mathrm{o}} \circ F_{\mathrm{e}}^{\mathrm{op}}$ *of functors* $S_{\mathrm{e}}^{\mathrm{op}} \to \mathbf{Type}\text{-}\mathbf{Alg}$.

5.11. DEFINITION. *Define the* category of schemas, *denoted* $\mathbf{Schema}$, *to have database schemas as objects and schema mappings as morphisms.*

5.12. REMARK. From the universal property of collages (Definition 2.16) in $\mathbb{P}\mathbf{rof}$, it follows easily that a schema mapping $\mathbf{F}\colon \mathbf{S} \to \mathbf{T}$ is equivalently a functor $\widetilde{F}$ over $\mathbf{2}$ between their collages, as in the left-hand diagram

$$
\begin{array}{ccc}
\widetilde{S} \xrightarrow{\ \widetilde{F}\ } \widetilde{T} & \qquad
\begin{array}{ccc}
S_{\mathrm{e}} & \xrightarrow{\ F_{\mathrm{e}}\ } & T_{\mathrm{e}} \\
{\scriptstyle i_S}\big\downarrow & & \big\downarrow{\scriptstyle i_T} \\
\widetilde{S} & \xrightarrow[\ \widetilde{F}\ ]{} & \widetilde{T}
\end{array}
& \qquad
\begin{array}{ccc}
\mathbf{Type} & =\!=\!= & \mathbf{Type} \\
{\scriptstyle i_{\mathbf{T}}}\big\downarrow & & \big\downarrow{\scriptstyle i_{\mathbf{T}}} \\
\widetilde{S} & \xrightarrow[\ \widetilde{F}\ ]{} & \widetilde{T}
\end{array}
\end{array} \tag{24}
$$

such that the middle and right-hand diagrams are the pullbacks of the left-hand diagram along the two maps $\mathbf{1} \to \mathbf{2}$; see (13) and Remark 5.3. By definition, a schema mapping acts as identity on the $\mathbf{Type}$-side of the collages.

5.13. EXAMPLE. Consider the schema presentation given by the following graph, attributes, and equations:


(25)
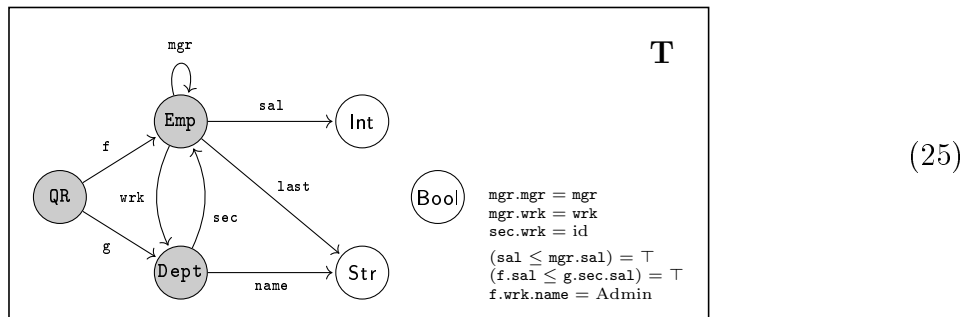
The schema **T** which it presents, includes **S** of Example 5.8. In addition it has a new entity QR — named for its eventual role as a query result table in Section 9 — as well as two new edges f, g, and two new observable equations

$$q : \text{QR} \vdash (q.\text{f.sal} \le q.\text{g.sec.sal}) = \top, \quad q : \text{QR} \vdash q.\text{f.wrk.name} = \text{Admin}. \tag{26}$$

Thus we have a schema inclusion $\mathbf{G} \colon \mathbf{S} \to \mathbf{T}$, which of course restricts to identity on the **Type**-side by definition.

5.14. EXAMPLE. We will now describe another schema mapping, with codomain the above schema **T**. We will again do so in terms of presentations:
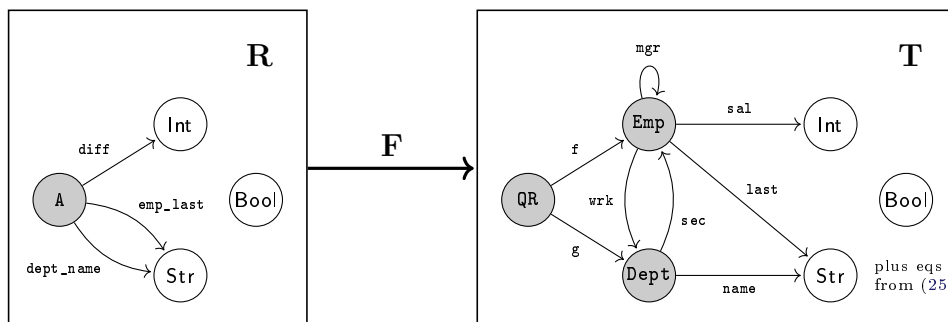
The schema **R** has a terminal entity category $R_{\text{e}} = \{\text{A}\}$, along with three generating attributes — namely diff, emp_last, and dept_name — from the unique object to the base **Type**-sorts Str, Int. Schema **T** has six equations, whereas **R** has none.

The schema mapping $\mathbf{F} \colon \mathbf{R} \to \mathbf{T}$ viewed as a functor $\widetilde{F} \colon \widetilde{R} \to \widetilde{T}$, is defined to map the unique object $\text{A} \in R_{\text{e}}$ to $\text{QR} \in T_{\text{e}}$ on the entity side, and to map the three attributes to the following observables in **T**:

$$\text{diff} \mapsto \text{g.sec.sal} - \text{f.sal} \quad \text{emp\_last} \mapsto \text{f.last} \quad \text{dept\_name} \mapsto \text{g.name}$$

Since on the type side it is identity and **R** has no equations, there is nothing more to check; we have defined a schema mapping. This choice will be justified by Remark 9.6.

Emp	last	wrk	mgr	sal
e1	Gauss	d3	e1	250
e2	Noether	d2	e4	200
e3	Einstein	d1	e3	300
e4	Turing	d2	e4	400
e5	Newton	d3	e1	100
e6	Euclid	d2	e7	150
e7	Hypatia	d2	e7	x

Dept	name	sec
d1	HR	e3
d2	Admin	e6
d3	IT	e5

Figure 4: Example of an **S**-instance **J**.

6. Algebraic database instances

6.1. INSTANCES AND TRANSFORMS. Given a database schema **S**, which is a conceptual layout of entities and their attributes (see Definition 5.2), we are ready to assign each entity a table full of data laid out according to the schema. Such an assignment is called an *instance* on **S**; it is a set-valued functor (copresheaf) of a certain form. Morphisms between instances are often called (attribute-preserving) database homomorphisms [AHV95], but we call them *transforms* because they are nothing more than natural transformations.

6.2. DEFINITION. *Let* **S** *be a database schema,* $\widetilde{S}$ *its collage, and* $i_{\mathbf{T}} \colon \mathbf{Type} \to \widetilde{S}$ *the inclusion of the type side (see Remark 5.3). An* **S**-*instance* **I** *is a functor* $I \colon \widetilde{S} \to \mathbf{Set}$ *such that the restriction* $I_{\mathrm{t}} := I \circ i_{\mathbf{T}}$ *preserves finite products, i.e.* $I_{\mathrm{t}} \colon \mathbf{Type} \to \mathbf{Set}$ *is a* **Type**-*algebra.*

Define the category of **S**-*instances, denoted* **S-Inst**, *to be the full subcategory of the functor category* $[\widetilde{S}, \mathbf{Set}]$ *spanned by the* **S**-*instances. A morphism* $\boldsymbol{\alpha} \colon \mathbf{I} \to \mathbf{J}$ *of instances is called a* transform.

6.3. EXAMPLE. Recall the schema **S** generated by the presentation of Example 5.8, which had employees and departments as entities, edges and attributes such as manager and salary, and equations such as an employee's salary must be less than that of his or her manager. A summary of an **S**-instance **J** is displayed in Fig. 4, with one table for each entity in **S**, and with a column for each edge and attribute.

All data required to determine an instance is encapsulated in the above two tables (image of the entity side and the attributes) along with a choice of **Type**-algebra, which is generally infinite. Here, the **Type**-algebra must include not only constants, but also all terms using the indeterminate $x : \mathsf{Int}$, which expresses Hypatia's unknown salary. Moreover, the equation $e : \mathtt{Emp} \vdash (e.\mathtt{sal} \leq e.\mathtt{mgr.sal}) = \top$ in the presentation of **S** implies that the terms $150 \leq x$ and $\top$ must be equal in $J(\mathsf{Bool})$ (by letting $e = e6$).

Explicitly, we can define the functor $J \colon \widetilde{S} \to \mathbf{Set}$ as follows: the restriction $J_{\mathrm{t}} = J \circ i_{\mathbf{T}}$ to **Type** is the presented type algebra

$$J_{\mathrm{t}} \cong \kappa[x : \mathsf{Int}]/(150 \leq x = \top),$$

and J is defined on entities by the following sets:

$$J(\texttt{Emp}) = \{e1, e2, e3, e4, e5, e6, e7\}, \quad J(\texttt{Dept}) = \{d1, d2, d3\},$$

and on edges and attributes by functions as shown in the table, e.g.

$$J(\texttt{wrk})\colon J(\texttt{Emp}) \to J(\texttt{Dept}) \quad \text{by} \quad e1 \mapsto d3, \ldots, e7 \mapsto d2$$
$$J(\texttt{name})\colon J(\texttt{Dept}) \to J(\texttt{Str}) \quad \text{by} \quad d1 \mapsto \mathrm{HR}, \ldots, d3 \mapsto \mathrm{IT}$$

6.4. EXAMPLE. Let $\mathbf{S}$ be as in Example 5.8. Consider another $\mathbf{S}$-instance $\bar{\mathbf{J}}$, which is the same as $\mathbf{J}$ except that $e7$ is removed from $\bar{J}(\texttt{Emp})$ and the restriction $\bar{J}_\mathrm{t} = \bar{J} \circ i_\mathbf{T}$ to $\mathbf{Type}$ is just κ, the algebra of constants as in Example 3.4. We will have use for both $\mathbf{J}$ and $\bar{\mathbf{J}}$ later.

6.5. DEFINITION. *We refer to instances whose* $\mathbf{Type}$*-algebra is initial, i.e.* $I_\mathrm{t} = \kappa$*, as* ground instances. *So* $\mathbf{J}$ *from Example 6.3 is not a ground instance but* $\bar{\mathbf{J}}$ *from Example 6.4 is. If a* $\mathbf{Type}$*-algebra is presented by generators and relations, generators (such as the indeterminate value 'x' of Example 6.3) are often referred to as* labelled nulls *or* Skolem variables *[AHV95].*

Even though it wasn't defined that way, the category of instances $\mathbf{S}$-$\mathbf{Inst}$ can be seen to be the category of algebras for an algebraic theory. Proving this, as we do next, immediately gives us several nice properties of the category $\mathbf{S}$-$\mathbf{Inst}$.

6.6. PROPOSITION. *For any schema* $\mathbf{S}$*, the category of instances* $\mathbf{S}$-$\mathbf{Inst}$ *is equivalent to the category of algebras for an algebraic theory.*

PROOF. Recall from Definition 3.3 the category of algebras for a theory. We can consider $\widetilde{S}$ as a finite-product sketch, whose designated product cones are all finite products in $\mathbf{Type}$. Then a model of this sketch is a functor $\widetilde{S} \to \mathbf{Set}$ which preserves finite products in $\mathbf{Type}$, i.e. an instance of $\mathbf{S}$. The category of models for any finite product sketch is equivalent to the category of algebras for an algebraic theory generated by the sketch; see e.g. [BW85, §4.3]. ∎

6.7. REMARK. When given a presentation (Ξ, E_Ξ) for a schema $\mathbf{S}$, as in Definition 5.6, we can make Proposition 6.6 much more concrete: combining Proposition 4.47 and Definition 6.2, it follows that there is an equivalence of categories $\mathbf{S}$-$\mathbf{Inst} \simeq (\mathbf{Cxt}_{\widetilde{\Xi}}/E_{\widetilde{\Xi}})$-$\mathbf{Alg}$.

6.8. COROLLARY. *For any schema* $\mathbf{S}$*, the category of instances* $\mathbf{S}$-$\mathbf{Inst}$ *has all small colimits.*

PROOF. This follows from 6.6 and 3.5. ∎

6.9. REMARK. As in Warning 3.6, we note that colimits in **S-Inst** do not always agree with the pointwise colimits in $[\widetilde{S}, \mathbf{Set}]$, which can make them difficult to work with. However, the following simple observation is sometimes useful:

Let $X \colon \mathscr{D} \to \mathbf{S\text{-}Inst}$ be a diagram and let $U(X)$ be its composite with the inclusion **S-Inst** $\to [\widetilde{S}, \mathbf{Set}]$. If the colimit $\mathrm{colim}(UX)$ in $[\widetilde{S}, \mathbf{Set}]$ lands in the subcategory **S-Inst** (i.e. preserves products in **Type**), then the natural map $\mathrm{colim}(UX) \to U(\mathrm{colim}\, X)$ is an isomorphism. In other words, in this case, the colimit $\mathrm{colim}\, X$ can be taken pointwise. (Note that this observation only uses the fact that **S-Inst** is a full subcategory of $[\widetilde{S}, \mathbf{Set}]$).

6.10. EXAMPLE. Let $\emptyset \in \mathbf{Schema}$ be the initial schema, i.e. the unique schema having an empty entity category. Then $\widetilde{\emptyset} \cong \mathbf{Type}$, thus there is an isomorphism of categories

$$\emptyset\text{-}\mathbf{Inst} \cong \mathbf{Type\text{-}Alg}.$$

6.11. REMARK. Notice that for any schema **S**, the Yoneda embedding $\mathbf{y} \colon \widetilde{S}^{\mathrm{op}} \to [\widetilde{S}, \mathbf{Set}]$ is product-preserving and hence factors through the forgetful functor **S-Inst** $\to [\widetilde{S}, \mathbf{Set}]$. The left factor $\widetilde{S}^{\mathrm{op}} \to \mathbf{S\text{-}Inst}$, which we also denote $\mathbf{y}$, is fully faithful. In particular, for any object $s \in \widetilde{S}$, the representable functor $\mathbf{y}(s) \colon \widetilde{S} \to \mathbf{Set}$ given by $\mathbf{y}(s)(x) = \widetilde{S}(s, x)$ is an instance, called the **S**-*instance represented by s*.

Because the functor **S-Inst** $\to [\widetilde{S}, \mathbf{Set}]$ is fully faithful, it follows that the embedding $\mathbf{y} \colon \widetilde{S}^{\mathrm{op}} \to \mathbf{S\text{-}Inst}$ is dense (see Remark 3.7). In particular, for any instance $I \in \mathbf{S\text{-}Inst}$, there is a canonical isomorphism of **S**-instances

$$I \cong \int^{s \in \widetilde{S}} I(s) \cdot \mathbf{y}(s)$$

which also follows from Remark 6.9.

6.12. PRESENTATIONS OF INSTANCES. Let (Ξ, E_Ξ) be a presentation of a schema S (Definition 5.6) and $\widetilde{\Xi}$ its associated algebraic signature, whose generated theory is the free product completion of its collage $\widetilde{S}$ (Proposition 4.47). By Remark 6.7, we can use presentations of algebras for a theory (Definition 4.32) to give presentations of **S**-instances.

6.13. DEFINITION. *Let Γ be a context over the above algebraic signature $\widetilde{\Xi}$. The free $(\mathbf{Cxt}_{\widetilde{\Xi}}/E_{\widetilde{\Xi}})$-algebra $\kappa[\Gamma]$ corresponds under the equivalence of Proposition 6.6 to an **S**-instance, which we denote $\langle \Gamma \rangle$, and call the* free **S**-*instance generated by Γ.*

*If E_Γ is a set of equations in context Γ, then we similarly write $\langle \Gamma \rangle/E_\Gamma$ for the **S**-instance corresponding to $\kappa[\Gamma]/E_\Gamma$, and call it the **S**-instance presented by (Γ, E_Γ). Concretely, $(\langle \Gamma \rangle/E_\Gamma)(s) = \{$terms in context Γ of type $s \in \widetilde{S}\}/\sim$, by Remark 4.27.*

6.14. EXAMPLE. For any object $s \in \widetilde{S}$, the representable instance $\mathbf{y}(s)$ as in Remark 6.11 is free, with one generator of type s, i.e. $\mathbf{y}(s) \cong \langle (x : s) \rangle$.

Emp	last	wrk	mgr	sal
e	e.last	e.wrk	e.mgr	e.sal
e.mgr	e.mgr.last	e.wrk	e.mgr	e.mgr.sal
d.sec	d.sec.last	d.sec.wrk $= d$	d.sec.mgr	d.sec.sal
d.sec.mgr	d.sec.mgr.last	d	d.sec.mgr	d.sec.mgr.sal
e.wrk.sec	e.wrk.sec.last	e.wrk	...	...
e.wrk.sec.mgr	...	...	...	...

Dept	name	sec
d	d.name	d.sec
e.wrk	Admin	e.wrk.sec

Figure 5: Example of a presented **S**-instance **I**.

6.15. REMARK. Similar to Remark 4.34, any given instance **I** has a canonical presentation, where for each $s \in \widetilde{S}$ and $x \in I(s)$ there is a generator $x : s$, and for each arrow $f : s \to s'$ in $\widetilde{S}$ with $y = I(f)(x)$, there is an equation $x : s \vdash x.f = y$. In this way, the presentation essentially records every entry of every column in every table.

For example, the canonical presentation of **J** from Example 6.3 has context and equations

$$\Gamma = (e1, \ldots, e7 : \text{Emp}, \, d1, d2, d3 : \text{Dept}, \, x : \text{Int}) \tag{27}$$
$$E = \{e1.\text{last} = \text{Gauss}, \, e1.\text{wrk} = d3, \, \ldots, \, d3.\text{sec} = e5\}$$

6.16. EXAMPLE. Let **S** be as in Example 5.8. We will now describe an **S**-instance **I** that is fairly different-looking than that in Example 6.3 or 6.4, in that the values of most of its attributes are non-constants. Instances like **I** play a central role in database queries (see Section 9).

We specify the instance **I** by means of a presentation $\mathbf{I} = \langle \Gamma \rangle / E_\Gamma$, where $\Gamma = (e : \text{Emp}, d : \text{Dept})$, and where E_Γ contains the two equations

$$\Gamma \vdash e.\text{wrk}.\text{name} = \text{Admin}$$
$$\Gamma \vdash (e.\text{sal} \leq d.\text{sec}.\text{sal}) = \top. \tag{28}$$

Thus for any entity or type $s \in \widetilde{S}$, the elements of $I(s)$ are the equivalence classes of terms $\Gamma \vdash t : s$ built out of edges and attributes from **S** and function symbols from **Type**, modulo the equations E_Γ as well as those from **S**.

We can picture this instance in the tables shown in Fig. 5. On types, I contains many terms, as in

$$I(\text{Str}) = \{e.\text{last}, e.\text{mgr}.\text{last}, d.\text{sec}.\text{last}, \ldots, d.\text{name}, \text{Admin}, \ldots, \text{aaBcZ}, \ldots\}$$
$$I(\text{Int}) = \{e.\text{sal}, e.\text{mgr}.\text{sal}, \ldots, -d.\text{sec}.\text{sal}, e.\text{sal} + e.\text{sal} + 1, \ldots, 28734, \ldots\}$$
$$I(\text{Bool}) = \{\text{eq}(e.\text{last}, d.\text{name}), \ldots, e.\text{mgr}.\text{sal} \leq d.\text{sec}.\text{sal}, \ldots, \top, \neg\top\}$$

Note, for example, that the value of $e.\mathtt{wrk}.\mathtt{name}$ in the table $\mathtt{Dept}$ has been replaced by 'Admin' because of an equation of $\mathbf{I}$ and that the value of $e.\mathtt{wrk}.\mathtt{sec}.\mathtt{wrk}$ has been replaced by $e.\mathtt{wrk}$ because of a path equation of $\mathbf{S}$ (see Example 5.8).

6.17. EXAMPLE. Having defined two $\mathbf{S}$-instances $\mathbf{J}$ and $\mathbf{I}$ in the examples 6.3 and 6.16 above, we will now explicitly describe the set $\mathbf{S}\text{-}\mathbf{Inst}(\mathbf{I}, \mathbf{J})$ of instance transforms between them.

By Remarks 4.34 and 6.7, the set of transforms $\mathbf{I} \to \mathbf{J}$ is equivalent to the set of morphisms of presentations from the presentation (Γ, E_Γ) of $\mathbf{I}$ to the canonical presentation of $\mathbf{J}$. Such a morphism of presentations is simply an assignment of an element $\epsilon \in J(\mathtt{Emp})$ to the generator $e : \mathtt{Emp}$ and an element $\delta \in J(\mathtt{Dept})$ to the generator $d : \mathtt{Dept}$, such that the two equations $\epsilon.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin}$ and $(\epsilon.\mathtt{sal} \leq \delta.\mathtt{sec}.\mathtt{sal}) = \top$ are true in $\mathbf{J}$.

Without the equations, there would be 21 assignments $(\epsilon, \delta) \in J(\mathtt{Emp}) \times J(\mathtt{Dept})$. It is easy to check that only three of those 21 satisfy the two equations: $(e6, d1)$, $(e2, d1)$, and $(e6, d2)$. For instance, the equation $e.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin}$ means we must have $\epsilon.\mathtt{wrk} = d2$. Similarly, the equation $\big((e.\mathtt{sal}) \leq d.\mathtt{sec}.\mathtt{sal}\big) = \top$ rules out several choices. For example, the assignment $(\epsilon, \delta) := (e7, d1)$ is invalid because we cannot deduce that $x \leq 300$ from any equations of $\mathbf{J}$ (where we only know that $150 \leq x$).

6.18. EXAMPLE. We will now consider transforms between two instances that are both presented. As usual, let $\mathbf{S}$ be the schema of Example 5.8, and let $\mathbf{I}$ be the instance from above, Example 6.16. We recall its presentation $(\Gamma_\mathbf{I}, E_\mathbf{I})$, as well as present a new $\mathbf{S}$-instance $\mathbf{I}'$:

$$\Gamma_{\mathbf{I}'} = \{e' : \mathtt{Emp}\} \qquad\qquad \Gamma_\mathbf{I} = \{e : \mathtt{Emp}, d : \mathtt{Dept}\}$$
$$E_{\mathbf{I}'} = \{e'.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin} \qquad\qquad E_\mathbf{I} = \{e.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin}$$
$$(e'.\mathtt{sal} \leq e'.\mathtt{wrk}.\mathtt{sec}.\mathtt{sal}) = \top\} \qquad\qquad (e.\mathtt{sal} \leq d.\mathtt{sec}.\mathtt{sal}) = \top\}$$

As in Definition 4.32, to give an instance transform $\beta \colon \mathbf{I}' \Rightarrow \mathbf{I}$, it is equivalent to give a context morphism $\Gamma_I \to \Gamma_{I'}$ (see Definition 4.11) in the opposite direction which respects the equations. In this case, there are only two which satisfy the equations: $[e' := e]$ and $[e' := e.\mathtt{wrk}.\mathtt{sec}]$.

6.19. DECOMPOSING INSTANCES. While Definition 6.2 is how we most often consider instances, it will sometimes be useful to consider their entity and attribute parts separately. Recall the left action $\otimes$ of Definition 3.11 on algebraic profunctors.

An instance $\mathbf{I}$ on a schema $\mathbf{S}$ is equivalently defined to be a tuple $(I_\mathrm{e}, I_\mathrm{t}, I_\mathrm{o})$, where
- $I_\mathrm{e} \colon \{*\} \twoheadrightarrow S_\mathrm{e}$ is a profunctor, called the *entity side* of $\mathbf{I}$,
- $I_\mathrm{t} \colon \{*\} \twoheadrightarrow \mathbf{Type}$ is an algebraic profunctor, called the *type side* of $\mathbf{I}$, and
- $I_\mathrm{o} \colon I_\mathrm{e} \otimes S_\mathrm{o} \to I_\mathrm{t}$ is a profunctor morphism, called the *values assignment* for $\mathbf{I}$:
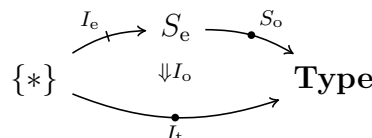
The functor $I \colon \widetilde{S} \to \mathbf{Set}$ of Definition 6.2, viewed as $I \colon \{*\} \rightarrowtail \widetilde{S}$, can then be uniquely recovered by the lax limit universal property of $\widetilde{S}$ spelled out in Remark 2.32, for $X = \{*\}$.

Note that the entity side $I_e \colon S_e \to \mathbf{Set}$ is just a copresheaf, the type side $I_t \colon \mathbf{Type} \to \mathbf{Set}$ is just a $\mathbf{Type}$-algebra, and the values assignment I_o is equivalent to a morphism $\int^{s \in S_e} I_e(s) \cdot S_o(s) \to I_t$ of $\mathbf{Type}$-algebras (where the coend is in $\mathbf{Type}\text{-}\mathbf{Alg}$, see Theorem 3.5). We could also obtain $I \colon \widetilde{S} \to \mathbf{Set}$ from collages universal property (11) in $\mathbb{P}\mathbf{rof}$.

Similarly, a transform $\mathbf{I} \to \mathbf{J}$ between instances can equivalently be defined in terms of separate entity and type components (α_e, α_t), where $\alpha_e \colon I_e \Rightarrow J_e$ and $\alpha_t \colon I_t \Rightarrow J_t$ are profunctor morphisms, satisfying the equation:

$$
\begin{array}{ccc}
\begin{array}{c}
\xymatrix{
& I_e & \\
\{*\} \ar@/^/[rr]^{} \ar@/_/[rr] & \Downarrow\alpha_e & S_e \ar[r]^{S_o} & \mathbf{Type} \\
& J_e & \Downarrow J_o
}
\end{array}
& = &
\begin{array}{c}
\xymatrix{
& I_e & S_e & S_o \\
\{*\} & \Downarrow I_o & & \mathbf{Type} \\
& I_t & \Downarrow\alpha_t & \\
& J_t &
}
\end{array}
\end{array}
$$

Given $\alpha \colon I \Rightarrow J \colon \widetilde{S} \to \mathbf{Set}$, the entity and type components α_e and α_t are simply the restrictions of α along the collage inclusions $i_S \colon S_e \to \widetilde{S} \leftarrow \mathbf{Type} \colon i_{\mathbf{T}}$. In the other direction, given α_e and α_t, one recovers α by the 2-dimensional part of the universal property of Remark 2.32.

7. The fundamental data migration functors

In this section, we describe functors that transfer instances from one schema to another. More specifically, we show how any schema mapping $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ induces a system of three adjoint functors

$$
\mathbf{T}\text{-}\mathbf{Inst}
\xymatrix{
\ar@/^1pc/[r]^{\Sigma_F} \ar@{}[r]|{\perp} \ar[r]|{\Delta_F} \ar@{}[r]|{\perp} \ar@/_1pc/[r]_{\Pi_F} &
}
\mathbf{S}\text{-}\mathbf{Inst}
$$

which we call *data migration functors*. They are related to the usual Kan extensions setting between categories of presheaves. Recall from Definition 5.10 that a schema mapping $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ is a functor $F_e \colon S_e \to T_e$ and a 2-cell F_o:

$$
\xymatrix{
S_e \ar[r]^{S_o} \ar[d]_{F_e} & \mathbf{Type} \ar@{=}[d] \\
T_e \ar[r]_{T_o} & \mathbf{Type}
}
\quad \Downarrow F_o
$$

7.1. DEFINITION. *Let* $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ *be a schema mapping, and let* $\widetilde{F} \colon \widetilde{S} \to \widetilde{T}$ *be the induced map on collages (Remark 5.12). We define a functor* $\Delta_F \colon \mathbf{T}\text{-}\mathbf{Inst} \to \mathbf{S}\text{-}\mathbf{Inst}$ *as follows:*

- *For any instance* $\mathbf{I}$ *of* $\mathbf{T}$*, define* $\Delta_F(\mathbf{I}) := I \circ \widetilde{F}$*. By* (24) *and Definition* 6.2*, the following diagram commutes:*

$$
\begin{array}{ccc}
 & \mathbf{Type} & \\
{\scriptstyle i_{\mathbf{T}}}\swarrow & {\scriptstyle i_{\mathbf{T}}}\downarrow & {\scriptstyle I_{\mathrm{t}}}\searrow \\
\widetilde{S} \xrightarrow{\ \widetilde{F}\ } & \widetilde{T} \xrightarrow{\ I\ } & \mathbf{Set}
\end{array}
$$

Thus $\Delta_F(\mathbf{I})_{\mathrm{t}} = I \circ \widetilde{F} \circ i_{\mathbf{T}} = I_{\mathrm{t}}$ *preserves products.*
- *For any* $\boldsymbol{\alpha}\colon \mathbf{I} \to \mathbf{J}$ *in* $\mathbf{T}$*-Inst, define* $\Delta_F(\alpha) = \alpha \circ \widetilde{F}$.

We call Δ_F *the* pullback functor *(along F).*

7.2. EXAMPLE. For any schema $\mathbf{S}$, the unique map $!\colon \emptyset \to \mathbf{S}$ from the initial schema (Example 6.10) induces a functor $\Delta_!\colon \mathbf{S}\text{-Inst} \to \mathbf{Type\text{-}Alg}$, denoted $\Delta_{!_S}$. For an instance $\mathbf{I}$ of $\mathbf{S}$, this functor returns the underlying $\mathbf{Type}$-algebra of the instance, $\Delta_{!_S}(\mathbf{I}) \cong I_{\mathrm{t}}$.

A schema mapping $\mathbf{F}$ can be considered as a map of finite product sketches; see Proposition 6.6. In general one does not expect the pullback functor Δ_F between the corresponding categories of algebras to have a right adjoint; for example, there is no 'cofree monoid' on a set. However, because $\mathbf{F}$ restricts to the identity on the $\mathbf{Type}$-side of the collage by Remark 5.12, we find that Δ_F does have a right adjoint, denoted Π_F, which we call the *right pushforward functor*.

7.3. PROPOSITION. *Let* $\mathbf{F}\colon \mathbf{S} \to \mathbf{T}$ *be a schema mapping. The right Kan extension* $\mathrm{Ran}_{\widetilde{F}}\colon [\widetilde{S}, \mathbf{Set}] \to [\widetilde{T}, \mathbf{Set}]$ *takes* $\mathbf{S}$*-instances to* $\mathbf{T}$*-instances, defining a right adjoint to* Δ_F,

$$\Delta_F\colon \mathbf{T}\text{-Inst} \leftrightarrows \mathbf{S}\text{-Inst} \colon \Pi_F$$

PROOF. Let $I\colon \widetilde{S} \to \mathbf{Set}$ be any functor. Then for an object $x \in \widetilde{T}$, the right Kan extension is given by

$$
\begin{aligned}
\Pi_F(I)(x) := (\mathrm{Ran}_{\widetilde{F}}I)(x) &\cong \left[\widetilde{T}, \mathbf{Set}\right]\left(\widetilde{T}(x, -), \mathrm{Ran}_{\widetilde{F}}(I)\right) \\
&\cong \left[\widetilde{S}, \mathbf{Set}\right]\left(\widetilde{T}(x, \widetilde{F}-), I\right) \\
&\cong \int_{s \in \widetilde{S}} I(s)^{\widetilde{T}(x, \widetilde{F}s)}.
\end{aligned}
\tag{29}
$$

We will show that this formula preserves the property of I_{t} being product-preserving. In fact, it preserves the $\mathbf{Type}$-algebra exactly, i.e. the diagram on the left commutes (up to natural isomorphism)

$$
\begin{array}{ccc}
\left[\widetilde{S}, \mathbf{Set}\right] \xrightarrow{\ \mathrm{Ran}_{\widetilde{F}}\ } \left[\widetilde{T}, \mathbf{Set}\right] & \qquad & \mathbf{Type} \xrightarrow{\ \mathrm{id}\ } \mathbf{Type} \\
{\scriptstyle -\circ i_{\mathbf{T}}}\searrow \quad \swarrow {\scriptstyle -\circ i_{\mathbf{T}}} & & {\scriptstyle i_{\mathbf{T}}}\downarrow \qquad \downarrow {\scriptstyle i_{\mathbf{T}}} \\
[\mathbf{Type}, \mathbf{Set}] & & \widetilde{S} \xrightarrow{\ \widetilde{F}\ } \widetilde{T}.
\end{array}
\tag{30}
$$

or equivalently, the pullback square on the right satisfies the Beck-Chevalley condition for right Kan extensions. The latter follows formally because the inclusion $i_{\mathbf{T}} \colon \mathbf{Type} \to \widetilde{T}$ is an opfibration, but we can easily check the commutativity of the left diagram directly: for any $\tau \in \mathbf{Type}$, (29) gives

$$(\mathrm{Ran}_{\widetilde{F}} I)(\tau) \cong \left[\widetilde{S}, \mathbf{Set}\right]\left(\widetilde{T}(\tau, \widetilde{F}-), I\right) \cong \left[\widetilde{S}, \mathbf{Set}\right]\left(\widetilde{S}(\tau, -), I\right) \cong I(\tau),$$

completing the proof.   ∎

We now define the *left pushforward functor*, denoted Σ_F.

7.4. PROPOSITION. *For any schema mapping* $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$, *the functor* Δ_F *has a left adjoint*

$$\Sigma_F \colon \mathbf{S\text{-}Inst} \leftrightarrows \mathbf{T\text{-}Inst} \colon \Delta_F.$$

If $\mathbf{I} \in \mathbf{S\text{-}Inst}$ *is an instance, then* $\Sigma_F(\mathbf{I})$ *is given by the following coend taken in* $\mathbf{T\text{-}Inst}$:

$$\Sigma_F(\mathbf{I}) \cong \int^{s \in \widetilde{S}} I(s) \cdot \mathbf{y}(\widetilde{F}s), \tag{31}$$

where $\mathbf{y}(\widetilde{F}s)$ *is the representable* $\mathbf{T}$-*instance* $\widetilde{T}(\widetilde{F}s, -)$, *see Remark* 6.11.
    In other words, Σ_F *is the left Kan extension*

$$\begin{array}{ccc} \widetilde{S}^{\mathrm{op}} & \xrightarrow{\ \ \widetilde{F}^{\mathrm{op}}\ \ } & \widetilde{T}^{\mathrm{op}} \\ {\scriptstyle\mathbf{y}}\Big\downarrow & & \Big\downarrow{\scriptstyle\mathbf{y}} \\ \mathbf{S\text{-}Inst} & \xrightarrow[\ \Sigma_F=\mathrm{Lan}_{\mathbf{y}}(\mathbf{y}\circ\widetilde{F})\ ]{} & \mathbf{T\text{-}Inst} \end{array} \tag{32}$$

and the above square in fact commutes.

PROOF. The coend exists because $\mathbf{T\text{-}Inst}$ is cocomplete (Corollary 6.8). It is simple to check that this defines a left adjoint to Δ_F:

$$\mathbf{T\text{-}Inst}\left(\int^{s\in\widetilde{S}} I(s) \cdot \mathbf{y}(\widetilde{F}^{\mathrm{op}}(s)), J\right) \cong \int_{s\in\widetilde{S}} \mathbf{T\text{-}Inst}\big(I(s)\cdot\mathbf{y}(\widetilde{F}^{\mathrm{op}}(s)), J\big)$$

$$\cong \int_{s\in\widetilde{S}} \mathbf{Set}\big(I(s), \mathbf{T\text{-}Inst}(\mathbf{y}(\widetilde{F}^{\mathrm{op}}(s)), J)\big)$$

$$\cong \int_{s\in\widetilde{S}} \mathbf{Set}\big(I(s), J(\widetilde{F}(s))\big)$$

$$\cong \left[\widetilde{S}, \mathbf{Set}\right](I, J\circ\widetilde{F}) = \mathbf{S\text{-}Inst}(I, \Delta_F(J)).$$

The square commutes since $\Sigma_F \colon \mathbf{S\text{-}Inst} \to \mathbf{T\text{-}Inst}$ is a pointwise Kan extension along the fully faithful $\mathbf{y}$.   ∎

7.5. REMARK. The coend (31) is not typically a pointwise colimit, as pointed out in Remark 6.9. Hence, unlike Π or Δ, given an object $t \in \widetilde{T}$ there is in general no explicit formula for computing the set $(\Sigma_F(I))(t)$.

However, obtaining the presentation of $\Sigma(I)$ from a presentation of I is almost trivial: if $\mathbf{I}$ is presented by a context $\Gamma = (x_1 : s_1, ..., x_n : s_n)$ and some equations, then $\Sigma_F(\mathbf{I})$ is presented by the context $F(\Gamma) = (x_1 : \widetilde{F}(s_1), ..., x_n : \widetilde{F}(s_n))$ and respective equations by applying $\widetilde{F}$ to edges, attributes of the term expressions.

7.6. REMARK. For any schema mapping $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$, one can check using (30) and Definition 7.1 that the functors Π_F and Δ_F preserve **Type**-algebras, in the sense that $(\Pi_F I)_t \cong I_t$ and $(\Delta_F J)_t \cong J_t$. This does not generally hold for Σ; in Proposition 7.12 we give a simple criterion for when it does.

7.7. EXAMPLE. We will give an example of the application of the left pushforward functor $\Sigma_H \colon \mathbf{S\text{-}Inst} \to \mathbf{L\text{-}Inst}$ on $\mathbf{J}$ from Example 6.3, for a schema mapping $\mathbf{H} \colon \mathbf{S} \to \mathbf{L}$ as follows:
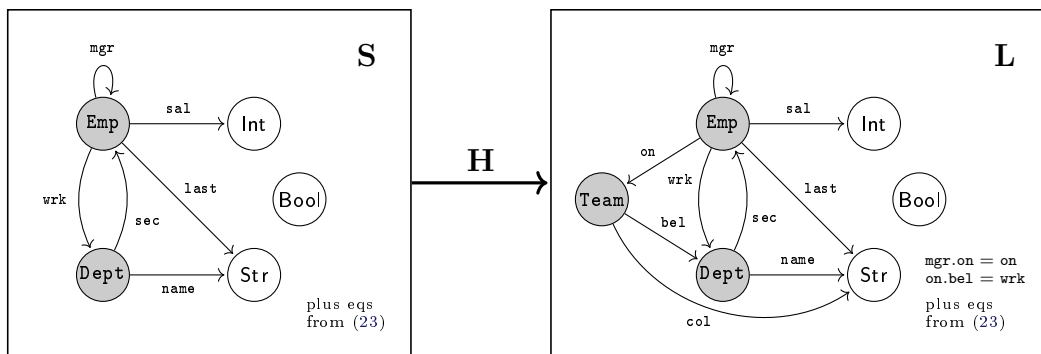
Schema $\mathbf{S}$ is as in Example 5.8, and schema $\mathbf{L}$ has a new entity 'Team', thought of as grouping employees into teams, which have a color-name and belong to some department. The two new equations ensure that an employee is on the same team as their manager and that their team belongs to their department.

The functor $\widetilde{H} \colon \widetilde{S} \to \widetilde{L}$ is an inclusion, preserving labels ($\widetilde{H}(\mathsf{Emp}) = \mathsf{Emp}$, etc.). Thus, by Remark 7.5, we find that the presentation of $\Sigma_H(\mathbf{J})$ is exactly that of $\mathbf{J}$, shown in (27), only now interpreted as a $\mathbf{L}$-instance presentation. To calculate the $\mathbf{L}$-instance it presents, one follows the explanation from Section 6.12 (as we explain briefly below) and finds that $\Sigma_H(\mathbf{J})$ is given by the tables shown in Fig. 6, where $t1, t2, t3, t4$ are freely generated terms. The **Type**-algebra of this instance is larger than that of $\mathbf{I}$; it includes, for example, new terms $t1.\mathtt{col}, \ldots, t4.\mathtt{col} : \mathsf{Str}$.

To calculate the set of rows in the $\mathtt{Team}$ table, following Definition 6.13 one freely adds a new team for each employee to be on, but quotients by setting each employees team equal to that of his or her manager, due to the equation $(\mathtt{mgr.on} = \mathtt{on}) : \mathtt{Team}$ in schema $\mathbf{L}$. Notice how we have one team belonging to HR and one team belonging to IT, but two teams belonging to Admin. This basically results from the freeness of the construction and the fact that there are two different managers, Turing and Hypatia, in

Emp	last	wrk	mgr	sal	on
e1	Gauss	*d3*	*e1*	250	*t1*
e2	Noether	*d2*	*e4*	200	*t2*
e3	Einstein	*d1*	*e3*	300	*t3*
e4	Turing	*d2*	*e4*	400	*t2*
e5	Newton	*d3*	*e1*	100	*t1*
e6	Euclid	*d2*	*e7*	150	*t4*
e7	Hypatia	*d2*	*e7*	x	*t4*

Team	col	bel
t1	*t1*.col	*d3*
t2	*t2*.col	*d2*
t3	*t3*.col	*d1*
t4	*t4*.col	*d2*

Dept	name	sec
d1	HR	*e3*
d2	Admin	*e6*
d3	IT	*e5*

Figure 6: The left pushforward instance, $\Sigma_H(\mathbf{J}) \in \mathbf{L\text{-}Inst}$.

Admin. The colors assigned to these teams are freely assigned as indeterminate string values (e.g. *t1*.col) in those cells.

7.8. EXAMPLE. Recall the schema mapping $\mathbf{G} \colon \mathbf{S} \to \mathbf{T}$ from Example 5.13, which is given by the inclusion of the $\mathbf{S}$-presentation (23) into the $\mathbf{T}$-presentation (25). We are going to describe the effect of the induced right pushforward functor $\Pi_G \colon \mathbf{S\text{-}Inst} \to \mathbf{T\text{-}Inst}$ on the $\mathbf{S}$-instance $\mathbf{J}$ of Example 6.3.

The $\mathbf{T}$-instance $\Pi_G(J) \colon \widetilde{T} \to \mathbf{Set}$ is given by an ordinary right Kan extension, as expressed by formula (29). Its $\mathbf{Type}$-algebra coincides with that of $\mathbf{J}$, namely it is the presented algebra $\kappa[x : \mathsf{Int}]/\sim$. Because $\mathbf{G}$ is of a particularly simple form, the only thing that remains to compute is $\Pi_G(J)(\mathtt{QR})$, which is the following subset of $J(\mathtt{Emp}) \times J(\mathtt{Dept})$:

QR	f	g
qr1	*e2*	*d1*
qr2	*e6*	*d1*
qr3	*e6*	*d2*

These three elements of the product are the ones that satisfy the supplementary equations of the presentation of $\mathbf{T}$ (i.e. $\mathtt{f.sal} \leq \mathtt{g.sec.sal} = \top$ and $\mathtt{f.wrk.name} = \mathrm{Admin}$). Its columns $\Pi_G(J)(\mathtt{f})$ and $\Pi_G(J)(\mathtt{g})$ represent the respective projections to $\Pi_G(J)(\mathtt{Emp}) = J(\mathtt{Emp})$ and $\Pi_G(J)(\mathtt{Dept}) = J(\mathtt{Dept})$. As usual, the names *qr1*, *qr2*, and *qr3* are not canonical; perhaps more canonical names would be (*e2*, *d1*), (*e6*, *d1*), and (*e6*, *d2*).

7.9. REMARK. One may notice that there is an isomorphism between the set $\Pi_G(J)(\mathtt{QR})$ from Example 7.8 and the set $\mathbf{S\text{-}Inst}(\mathbf{I}, \mathbf{J})$ from Example 6.17. The reason is that there is in fact an isomorphism of $\mathbf{S}$-instances $\widetilde{T}(\mathtt{QR}, \widetilde{G}-) \cong I$, as is most evident by observing the similarity between the defining equations (26) and (28).

7.10. EXAMPLE. Recall the schema mapping $\mathbf{F} \colon \mathbf{R} \to \mathbf{T}$ described in Example 5.14. Here we will discuss the pullback $\Delta_F(\mathbf{K})$, where $\mathbf{K} := \Pi_G(\mathbf{J})$ is computed in Example 7.8. Briefly, the table presentation of $\mathbf{K}$ consists of the tables $\mathtt{Emp}, \mathtt{Dept}$ as in Example 6.3 and $\mathtt{QR}$ as in 7.8, and its $\mathbf{Type}$-algebra is $\kappa[x : \mathsf{Int}]/\sim$.

The $\mathbf{R}$-instance $\Delta_F(K)\colon \widetilde{R} \to \mathbf{Set}$ is obtained by pre-composing with $\widetilde{F}$, as in Definition 7.1. It has the same $\mathbf{Type}$-algebra (Remark 7.6), and its one entity table is

A	emp_last	dept_name	diff
qr1	Noether	HR	100
qr2	Euclid	HR	150
qr3	Euclid	Admin	0

We conclude this section with some special cases for which Σ is nicely behaved.

7.11. A POINTWISE FORMULA FOR Σ. Given an arbitrary mapping $\mathbf{F}\colon \mathbf{S} \to \mathbf{T}$ and $\mathbf{S}$-instance $\mathbf{I}$, the formula for the functor $\Sigma_F(I)\colon \widetilde{T} \to \mathbf{Set}$ cannot be given pointwise on objects $t \in \widetilde{T}$. However, there is a special kind of schema mapping F for which we can write a pointwise formula for $\Sigma_F(\mathbf{I})$, namely those which induce a *discrete opfibration* on collages $\widetilde{F}\colon \widetilde{S} \to \widetilde{T}$. This occurs if and only if $\widetilde{F}$ arises via the Grothendieck construction applied to a functor $\partial F\colon \widetilde{T} \to \mathbf{Set}$, for which the composite $\partial F \circ i_{\mathbf{T}}\colon \mathbf{Type} \to \widetilde{T} \to \mathbf{Set}$ is terminal. Note that in this case, we have a bijection

$$\mathrm{Ob}\,\widetilde{S} \cong \left\{ (t,p) \;\middle|\; t \in \widetilde{T}, p \in \partial F(t) \right\} \cong \coprod_{t \in \widetilde{T}} \partial F(t).$$

One can show using ends, the adjunction $\Sigma \dashv \Delta$, and the fact that $\mathbf{S\text{-}Inst} \subseteq [\widetilde{S}, \mathbf{Set}]$ is fully faithful, that Σ_F is then given by the following pointwise formula:

$$\Sigma_F(I)(t) = \coprod_{p \in \partial F(t)} I(t,p).$$

In particular, Σ_F preserves $\mathbf{Type}$-algebras in this case, i.e. $\Sigma_F(I)(\tau) = I(\tau)$ for any $\tau \in \mathbf{Type}$.

It is easy to show that if $\widetilde{F}$ is a discrete opfibration, then F_{o} is cartesian, so the preservation of $\mathbf{Type}$-algebras can also be seen as a special case of the following result.

7.12. PROPOSITION. *The left pushforward Σ_F along a schema mapping $\mathbf{F} = (F_{\mathrm{e}}, F_{\mathrm{o}})\colon \mathbf{S} \to \mathbf{T}$ preserves type-algebras if and only if F_{o} is cartesian.*

PROOF SKETCH. Consider the commutative square in $\mathbf{Schema}$ shown here:

$$
\begin{array}{ccc}
 & \emptyset & \\
{}^{!_R}\swarrow & & \searrow{}^{!_S} \\
\mathbf{R} & \underset{\mathbf{F}}{\rightrightarrows} & \mathbf{S}
\end{array}
$$

By Example 7.2 it suffices to show that F_{o} is cartesian if and only if the restriction of the unit map $\Delta_{!_R}\eta\colon \Delta_{!_R} \to \Delta_{!_R}\Delta_F\Sigma_F = \Delta_{!_S}\Sigma_F$ coming from $\Sigma_F \dashv \Delta_F$ is an isomorphism. Both sides preserve colimits, so since $\mathbf{y}$ is dense, $\Delta_{!_R}\eta$ is an isomorphism if and only if the

components $\Delta_{!_R}(\eta_{\mathbf{y}(r)})$ are isomorphisms for any $r \in \widetilde{R}$. For $\tau \in \mathbf{Type}$, $\eta_{\mathbf{y}(\tau)}$ is always an isomorphism. For $r \in \widetilde{R}$, we have $\Delta_{!_R}(\mathbf{y}(r)) = R_{\mathrm{o}}(r)$ and $\Delta_{!_S}\Sigma_F(\mathbf{y}(r)) = S_{\mathrm{o}}(F_{\mathrm{e}}(r))$ by (32). It is not difficult to verify that $\Delta_{!_R}(\eta_{\mathbf{y}(r)}) \colon \Delta_{!_R}(\mathbf{y}(r)) \to \Delta_{!_S}\Sigma_F(\mathbf{y}(r))$ and the component $R_{\mathrm{o}}(r) \to S_{\mathrm{o}}(F_{\mathrm{e}}(r))$ of F_{o} at r agree, completing the proof. ∎

8. The double category $\mathbb{D}\mathbf{ata}$

In this section, we will introduce the notion of a *bimodule* between two schemas. We will see that bimodules generalize instances on a schema, as well as *queries*, which are the subject of Section 9. We will show that schemas, schema mappings, and bimodules together form an equipment, which we denote $\mathbb{D}\mathbf{ata}$. For database-style examples of material from this section, see Section 9.

8.1. RELEVANT TERMINOLOGY AND NOTATION. Recall that companions and conjoints in $\mathbb{P}\mathbf{rof}$ are given by representable profunctors, as explained in Example 2.12. Also recall from Definition 3.10 that a profunctor M whose codomain is an algebraic theory $\mathscr{T}$ is called algebraic if it is product-preserving on the right; it is denoted $M \colon \mathscr{C} \nrightarrow \mathscr{T}$. If $\mathbf{S}$ is a schema, then the functor $i_{\mathbf{T}} \colon \mathbf{Type} \to \widetilde{S}$ denotes the inclusion of $\mathbf{Type}$ into the collage (Remark 5.3).

8.2. BIMODULES BETWEEN SCHEMAS. Bimodules admit several equivalent definitions, and it is convenient to be able to switch between these definitions as best suits the task at hand. We will begin with the one which we use most often.

8.3. DEFINITION. *Let $\mathbf{R}$ and $\mathbf{S}$ be database schemas. A* bimodule $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ *is a functor* $M \colon \widetilde{R}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$ *such that the following diagram commutes:*

$$
\begin{array}{ccc}
\mathbf{Type}^{\mathrm{op}} & \xrightarrow{\;i_{\mathbf{T}}^{\mathrm{op}}\;} & \widetilde{S}^{\mathrm{op}} \\
{\scriptstyle i_{\mathbf{T}}^{\mathrm{op}}}\Big\downarrow & & \Big\downarrow{\scriptstyle \mathbf{y}} \\
\widetilde{R}^{\mathrm{op}} & \xrightarrow[\;M\;]{} & \mathbf{S}\text{-}\mathbf{Inst}
\end{array}
\tag{33}
$$

or succinctly, $M(\tau) = \mathbf{y}(\tau)$ for any $\tau \in \mathbf{Type}$.

A morphism of $(\mathbf{R}, \mathbf{S})$-bimodules $\phi \colon \mathbf{M} \to \mathbf{N}$ is a natural transformation $\phi \colon M \Rightarrow N$ that restricts to the identity on $\mathbf{Type}$. We denote by $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ the category of $(\mathbf{R}, \mathbf{S})$-bimodules.

8.4. REMARK. It is possible to give Definition 8.3 in a more symmetric form. A bimodule $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ is equivalently a profunctor $\widetilde{M} \colon \widetilde{R} \nrightarrow \widetilde{S}$ between collages such that:

- the composite profunctor $\widetilde{R} \xrightarrow{\widetilde{M}} \widetilde{S} \xrightarrow{\widetilde{i_{\mathbf{T}}}} \mathbf{Type}$ is algebraic, and
- the composite profunctor $\mathbf{Type} \xrightarrow{\widehat{i_{\mathbf{T}}}} \widetilde{R} \xrightarrow{\widetilde{M}} \widetilde{S}$ is isomorphic to the representable $\widehat{i_{\mathbf{T}}} \colon \mathbf{Type} \nrightarrow \widetilde{S}$.

A morphism of $(\mathbf{R}, \mathbf{S})$-bimodules $\phi \colon \mathbf{M} \to \mathbf{N}$ is equivalently a profunctor transformation $\widetilde{\phi} \colon \widetilde{M} \Rightarrow \widetilde{N}$ such that $\widehat{i_{\mathbf{T}}} \odot \widetilde{\phi} = \mathrm{id}_{\widehat{i_{\mathbf{T}}}}$.

While this formulation of bimodules may be useful for intuition, we will primarily use Definition 8.3 in this paper.

8.5. ADJOINTS Λ AND Γ. Considering a bimodule $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ as a functor $\widetilde{R}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$, we can apply the left Kan extension along the Yoneda embedding $\widetilde{R}^{\mathrm{op}} \to \mathbf{R}\text{-}\mathbf{Inst}$; see Remark 6.11. The result is denoted $\Lambda_M := \mathrm{Lan}_Y(M)$,

$$
\begin{array}{ccc}
\widetilde{R}^{\mathrm{op}} & \xrightarrow{\;\;M\;\;} & \mathbf{S}\text{-}\mathbf{Inst} \\
{\scriptstyle \mathbf{y}}\Big\downarrow & \nearrow & \\
\mathbf{R}\text{-}\mathbf{Inst} & {\scriptstyle \Lambda_M} &
\end{array}
\tag{34}
$$

Since $\mathbf{S}\text{-}\mathbf{Inst}$ is cocomplete (Corollary 6.8), we can express this using the Kan extension formula (cf. (2))

$$
\begin{aligned}
\Lambda_M(I) &= \int^{r \in \widetilde{R}} \mathbf{R}\text{-}\mathbf{Inst}\big(\mathbf{y}(r), I\big) \cdot M(r) \\
&\cong \int^{r \in \widetilde{R}} I(r) \cdot M(r)
\end{aligned}
\tag{35}
$$

where $\cdot$ is the $\mathbf{Set}$-theoretic copower on $\mathbf{S}\text{-}\mathbf{Inst}$. Because the Yoneda embedding is fully faithful, it follows that this Kan extension really is an extension, i.e. (34) commutes. It also follows that Λ_M "preserves types," that is, that the following diagram commutes:

$$
\begin{array}{ccc}
 & \mathbf{Type}^{\mathrm{op}} & \\
{\scriptstyle \mathbf{y}}\swarrow & & \searrow{\scriptstyle \mathbf{y}} \\
\mathbf{R}\text{-}\mathbf{Inst} & \xrightarrow[\;\;\Lambda_M\;\;]{} & \mathbf{S}\text{-}\mathbf{Inst}
\end{array}
\tag{36}
$$

A bimodule $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ also determines a functor in the other direction,

$$
\begin{aligned}
\Gamma_M \colon \mathbf{S}\text{-}\mathbf{Inst} &\xrightarrow{\qquad} \mathbf{R}\text{-}\mathbf{Inst} \\
J &\longmapsto \mathbf{S}\text{-}\mathbf{Inst}(M(-), J).
\end{aligned}
\tag{37}
$$

The condition (33) on M implies that for any object $\tau \in \mathbf{Type}$,

$$
(\Gamma_M J)(\tau) = \mathbf{S}\text{-}\mathbf{Inst}(M(\tau), J) = \mathbf{S}\text{-}\mathbf{Inst}(\mathbf{y}(\tau), J) = J(\tau)
\tag{38}
$$

from which it easily follows that $\Gamma_M(J)$ preserves products of types, hence defines an object in $\mathbf{R}\text{-}\mathbf{Inst}$. We thus say that Γ "preserves type-algebras", in the sense that the following diagram commutes:

$$
\begin{array}{ccc}
\mathbf{S}\text{-}\mathbf{Inst} & \xrightarrow{\;\;\Gamma_M\;\;} & \mathbf{R}\text{-}\mathbf{Inst} \\
{\scriptstyle U}\searrow & & \swarrow{\scriptstyle U} \\
 & \mathbf{Type}\text{-}\mathbf{Alg} &
\end{array}
$$

8.6. PROPOSITION. *For any bimodule* $\mathbf{M} \colon \mathbf{R} \twoheadrightarrow \mathbf{S}$, *the functor* Λ_M *is left adjoint to* Γ_M.

PROOF. This is simply a calculation:

$$\mathbf{S}\text{-}\mathbf{Inst}(\Lambda_M I, J) = \mathbf{S}\text{-}\mathbf{Inst}\left(\int^{r \in \widetilde{R}} I(r) \cdot M(r), J\right)$$

$$\cong \int_{r \in \widetilde{R}} \mathbf{S}\text{-}\mathbf{Inst}\big(I(r) \cdot M(r), J\big)$$

$$\cong \int_{r \in \widetilde{R}} \mathbf{Set}\big(I(r), \mathbf{S}\text{-}\mathbf{Inst}(M(r), J)\big)$$

$$= \int_{r \in \widetilde{R}} \mathbf{Set}\big(I(r), (\Gamma_M J)(r)\big) \cong \mathbf{R}\text{-}\mathbf{Inst}(I, \Gamma_M J).$$

The first isomorphism follows because homs take colimits in their first variable to limits, while the second is the definition of copower. ∎

8.7. LEMMA. *We collect here several easy but useful properties of* Λ:
1. *For any schema* $\mathbf{S}$, *there is an isomorphism of functors* $\Lambda_{\mathbf{y}} \cong \mathrm{id}_{\mathbf{S}\text{-}\mathbf{Inst}}$.
2. *For any bimodule* $M \colon \widetilde{R}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$ *and any left adjoint* $L \colon \mathbf{S}\text{-}\mathbf{Inst} \to \mathbf{T}\text{-}\mathbf{Inst}$, *there is an isomorphism of functors* $\Lambda_{L \circ M} \cong L \circ \Lambda_M$. *In particular,*
3. *For any bimodules* $\mathbf{M} \colon \mathbf{R} \twoheadrightarrow \mathbf{S}$ *and* $\mathbf{N} \colon \mathbf{S} \twoheadrightarrow \mathbf{T}$, *there is an isomorphism of functors* $\Lambda_{\Lambda_N \circ M} \cong \Lambda_N \circ \Lambda_M$.

PROOF. Property 1 is simply the fact that $\mathbf{y}$ is dense (see Remark 6.11), while property 2 is the fact that left adjoints preserve colimits, hence preserve pointwise left Kan extensions. Finally, property 3 follows from property 2 using Proposition 8.6. ∎

8.8. REMARK. The $\Lambda_M \dashv \Gamma_M$ adjunction is an instance of the general geometric realization/nerve adjunction $\mathbf{Set}^{\mathscr{S}^{\mathrm{op}}} \leftrightarrows \mathscr{C}$ induced by a functor $F \colon \mathscr{S} \to \mathscr{C}$ into a cocomplete category $\mathscr{C}$ (see e.g. [Lei04, pp. 244–245] or [nLab]). In this case, F is the functor $M \colon \widetilde{R}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$. The conditions in Definition 8.3 guarantee that the nerve functor lands in the full subcategory $\mathbf{R}\text{-}\mathbf{Inst} \subseteq [\widetilde{R}, \mathbf{Set}]$.

8.9. EQUIVALENT DEFINITIONS OF BIMODULES. In Theorem 8.10 we give five equivalent definitions of bimodules, and we will give a few others throughout the section, e.g. in Propositions 8.20 and 8.24 and Corollary 8.35. The ones we discuss here are aligned with the analogy presented in Section 2.2, by which profunctors between categories and linear transformations between vector spaces can be related. The only complication here is that all of our structures must deal carefully with the algebraic theory $\mathbf{Type}$, as we now make explicit.

Consider the coslice 2-category $\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at}$. An object is a pair $(\mathscr{C}, F)$, where $\mathscr{C} \in \mathcal{C}\mathbf{at}$ is a category and $F \colon \mathbf{Type}^{\mathrm{op}} \to \mathscr{C}$ is a functor; a morphism $(\mathscr{C}, F) \to (\mathscr{D}, G)$ is a functor $H \colon \mathscr{C} \to \mathscr{D}$ such that $H \circ F = G$; and a 2-cell $H \to H'$, where $H, H' \colon (\mathscr{C}, F) \to (\mathscr{D}, G)$, is a natural transformation $\alpha \colon H \Rightarrow H'$ such that $\alpha F = \mathrm{id}_G$.

For any schema **S**, both **S-Inst** and $\widetilde{S}^{\mathrm{op}}$ can be considered objects in $\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at}$ (via **y** and $i_{\mathbf{T}}^{\mathrm{op}}$). Similarly, **S-Inst** can be considered an object in the slice 2-category $\mathcal{C}\mathbf{at}/\mathbf{Type\text{-}Alg}$, where the functor **S-Inst** $\to$ **Type-Alg** simply sends an instance $\widetilde{S} \to$ **Set** to its restriction along the inclusion $i_{\mathbf{T}} \colon \mathbf{Type} \to \widetilde{S}$.

8.10. THEOREM. *Let* **R** *and* **S** *be schemas. The following are equivalent:*
1. *The category* $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ *of bimodules* $\mathbf{R} \twoheadrightarrow \mathbf{S}$.
2. *The category* $(\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at})(\widetilde{R}^{\mathrm{op}}, \mathbf{S\text{-}Inst})$.
3. *The category of profunctors* $\widetilde{R} \twoheadrightarrow \widetilde{S}$ *satisfying the conditions of Remark 8.4.*
4. *The category* $\mathcal{L}\mathbf{Adj}_{\mathbf{Type}}(\mathbf{R\text{-}Inst}, \mathbf{S\text{-}Inst})$, *which we define to be the full subcategory of* $(\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at})(\mathbf{R\text{-}Inst}, \mathbf{S\text{-}Inst})$ *spanned by left adjoint functors.*
5. *The category* $\mathcal{R}\mathbf{Adj}_{\mathbf{Type}}(\mathbf{S\text{-}Inst}, \mathbf{R\text{-}Inst})^{\mathrm{op}}$, *whose opposite is defined to be the full subcategory of* $(\mathcal{C}\mathbf{at}/\mathbf{Type\text{-}Alg})(\mathbf{S\text{-}Inst}, \mathbf{R\text{-}Inst})$ *spanned by right adjoint functors.*

PROOF. 1 and 2 are equivalent by Definition 8.3, and it is easy to check the equivalence between 1 and 3.

For the equivalence of 2 and 4, consider the functor

$$(- \circ \mathbf{y}) \colon \mathcal{L}\mathbf{Adj}_{\mathbf{Type}}(\mathbf{R\text{-}Inst}, \mathbf{S\text{-}Inst}) \to (\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at})(\widetilde{R}^{\mathrm{op}}, \mathbf{S\text{-}Inst}).$$

Its inverse is Λ_{-}, the left Kan extension along $\mathbf{y} \colon \widetilde{R}^{\mathrm{op}} \to \mathbf{R\text{-}Inst}$, which lands in $\mathcal{L}\mathbf{Adj}_{\mathbf{Type}}(\mathbf{R\text{-}Inst}, \mathbf{S\text{-}Inst})$ by (36) and Proposition 8.6. To see that these are inverses, note that by commutative (34) we have $\Lambda_M\big(\mathbf{y}(r)\big) = M(r)$. For the other direction, we have by Lemma 8.7

$$\Lambda_{L \circ \mathbf{y}} \cong L \circ \Lambda_{\mathbf{y}} \cong L \circ \mathrm{id}_{\mathbf{S\text{-}Inst}} = L.$$

Finally, we show that 4 and 5 are equivalent. The equivalence $\mathcal{L}\mathbf{Adj}(\mathbf{R\text{-}Inst}, \mathbf{S\text{-}Inst}) \simeq \mathcal{R}\mathbf{Adj}(\mathbf{S\text{-}Inst}, \mathbf{R\text{-}Inst})^{\mathrm{op}}$ is standard, so we only need to show that this equivalence respects the restrictions concerning **Type**. In one direction, if $L \colon \mathbf{R\text{-}Inst} \to \mathbf{S\text{-}Inst}$ is a left adjoint satisfying (36), then we check that the right adjoint G of L satisfies (38):

$$\begin{aligned}
(GJ)(\tau) &\cong \mathbf{R\text{-}Inst}\big(\mathbf{y}(\tau), GJ\big) \\
&\cong \mathbf{S\text{-}Inst}\Big(L\big(\mathbf{y}(\tau)\big), J\Big) \\
&\cong \mathbf{S\text{-}Inst}\big(\mathbf{y}(\tau), J\big) \\
&\cong J(\tau).
\end{aligned}$$

Conversely, if $G \colon \mathbf{S\text{-}Inst} \to \mathbf{R\text{-}Inst}$ is a right adjoint satisfying (38), then

$$\begin{aligned}
\mathbf{S\text{-}Inst}\Big(L\big(\mathbf{y}(\tau)\big), J\Big) &\cong \mathbf{R\text{-}Inst}\big(\mathbf{y}(\tau), RJ\big) \\
&\cong (GJ)(\tau) \\
&\cong J(\tau) \\
&\cong \mathbf{S\text{-}Inst}\big(\mathbf{y}(\tau), J\big),
\end{aligned}$$

hence by the Yoneda lemma, $L\big(\mathbf{y}(\tau)\big) \cong \mathbf{y}(\tau)$, for any $\tau \in \mathbf{Type}$. ∎

8.11. PROPOSITION. *For any schemas $\mathbf{R}$ and $\mathbf{S}$, the category $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ has finite colimits.*

PROOF. The initial object of $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ is given by the left Kan extension of the Yoneda embedding $\mathbf{Type}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$ along the collage inclusion $\mathbf{Type}^{\mathrm{op}} \to \widetilde{R}^{\mathrm{op}}$. Concretely, the initial bimodule $\mathbf{0}$ can be described by cases:

$$0(r, s) = \begin{cases} \widetilde{R}(r, s) & \text{if } s \text{ is a type} \\ \emptyset & \text{otherwise.} \end{cases}$$

To complete the proof, we need to show that $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ has pushouts. By Theorem 8.10, $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}} \simeq (\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at})(\widetilde{R}^{\mathrm{op}}, \mathbf{S}\text{-}\mathbf{Inst})$, and by Corollary 6.8, $\mathbf{S}\text{-}\mathbf{Inst}$ is cocomplete. Let us fix a choice of pushouts in $\mathbf{S}\text{-}\mathbf{Inst}$, such that the chosen pushout of the constant span on an instance I is I. Then it is easy to check that $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$ is closed under the induced chosen pointwise pushouts in $\mathcal{C}\mathbf{at}(\widetilde{R}^{\mathrm{op}}, \mathbf{S}\text{-}\mathbf{Inst})$, and that these are in fact pushouts in the subcategory $(\mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at})(\widetilde{R}^{\mathrm{op}}, \mathbf{S}\text{-}\mathbf{Inst})$. ∎

8.12. THE EQUIPMENT $\mathbb{D}\mathbf{ata}$. We are now ready to assemble schemas, schema morphisms, and bimodules into a single double category $\mathbb{D}\mathbf{ata}$, which we define in Definition 8.13, and which we show to be an equipment in Proposition 8.14. In order to define the double category structure, we will need the easy notion of restriction of bimodules along schema morphisms.

Suppose we have a bimodule $\mathbf{N} \colon \mathbf{R}' \nrightarrow \mathbf{S}'$, and schema mappings $\mathbf{F} \colon \mathbf{R} \to \mathbf{R}'$ and $\mathbf{G} \colon \mathbf{S} \to \mathbf{S}'$. Thinking of $\mathbf{N}$ as a functor $N \colon \widetilde{R}'^{\mathrm{op}} \to \mathbf{S}'\text{-}\mathbf{Inst}$ as in Definition 8.3, we can form the bottom composite

$$
\begin{array}{ccc}
\widetilde{R}^{\mathrm{op}} & \dashrightarrow{\ _{F}N_{G}\ } & \mathbf{S}\text{-}\mathbf{Inst} \\
{\scriptstyle \widetilde{F}^{\mathrm{op}}}\downarrow & & \uparrow{\scriptstyle \Delta_{G}} \\
(\widetilde{R}')^{\mathrm{op}} & \xrightarrow[\ N\ ]{} & \mathbf{S}'\text{-}\mathbf{Inst}
\end{array}
$$

and define a bimodule $_{\mathbf{F}}\mathbf{N}_{\mathbf{G}} \colon \mathbf{R} \nrightarrow \mathbf{S}$ so that the square commutes. This construction defines a functor $_{F}\mathrm{Bimod}_{G} \colon {}_{\mathbf{R}'}\mathrm{Bimod}_{\mathbf{S}'} \to {}_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$. By computing the composite $\Delta_{F} \circ N \circ \widetilde{F}^{\mathrm{op}}$ on objects, it easily follows that $\big(_{F}N_{G}(r)\big)(s) = \widetilde{N}(\widetilde{F}r, \widetilde{G}s)$ for any $r \in \widetilde{R}$ and $s \in \widetilde{S}$, where $\mathbf{N}$ is viewed as $\widetilde{N} \colon \widetilde{R}' \nrightarrow \widetilde{S}'$. This is relevant to Remark 8.15.

8.13. DEFINITION. *We define the double category $\mathbb{D}\mathbf{ata}$ as follows: the objects of $\mathbb{D}\mathbf{ata}$ are schemas, the vertical morphisms are schema mappings, and the horizontal morphisms are bimodules. We define a 2-cell of the form*

$$
\begin{array}{ccc}
\mathbf{R} & \xrightarrow{\ \mathbf{M}\ } & \mathbf{S} \\
{\scriptstyle \mathbf{F}}\downarrow & \Downarrow\boldsymbol{\theta} & \downarrow{\scriptstyle \mathbf{G}} \\
\mathbf{R}' & \xrightarrow[\ \mathbf{N}\ ]{} & \mathbf{S}'
\end{array}
\tag{39}
$$

to be a natural transformation $\theta \colon M \to \Delta_G \circ N \circ \widetilde{F}^{\mathrm{op}}$:

$$
\begin{array}{ccc}
\widetilde{R}^{\mathrm{op}} & \xrightarrow{\ M\ } & \mathbf{S}\text{-}\mathbf{Inst} \\
{\scriptstyle \widetilde{F}^{\mathrm{op}}} \downarrow & \Downarrow \theta & \uparrow {\scriptstyle \Delta_G} \\
\widetilde{R'}^{\,\mathrm{op}} & \xrightarrow[\ N\ ]{} & \mathbf{S'}\text{-}\mathbf{Inst},
\end{array}
$$

i.e. a morphism $\boldsymbol{\theta} \in {}_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}(\mathbf{M}, {}_{\mathbf{F}}\mathbf{N}_{\mathbf{G}})$. *Equivalently, it is a 2-cell* $\widetilde{\theta}\colon \widetilde{M} \Rightarrow \widetilde{N}$ *in* $\mathbb{P}\mathrm{rof}$ *with frames* $\mathsf{L}(\widetilde{\theta}) = \widetilde{F}$ *and* $\mathsf{R}(\widetilde{\theta}) = \widetilde{G}$, *and which has identity components on* $r \in \mathbf{Type}$.
   Given bimodules $\mathbf{M}\colon \mathbf{R} \nrightarrow \mathbf{S}$ *and* $\mathbf{N}\colon \mathbf{S} \nrightarrow \mathbf{T}$, *we define their composite* $\mathbf{M} \odot \mathbf{N}$ *by*

$$
M \odot N \coloneqq \Lambda_N \circ M \colon \widetilde{R}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst} \to \mathbf{T}\text{-}\mathbf{Inst}. \tag{40}
$$

where Λ_N *is as defined in* (35). *The unit bimodule* $\mathsf{U}_{\mathbf{R}}\colon \mathbf{R} \nrightarrow \mathbf{R}$ *for any schema* $\mathbf{R}$ *is given by the Yoneda embedding* $\mathbf{y}\colon \widetilde{R}^{\mathrm{op}} \to \mathbf{R}\text{-}\mathbf{Inst}$, *since* $\Lambda_M \circ \mathbf{y} \cong M$ *by* (34). *It corresponds to the unit in* $\mathbb{P}\mathrm{rof}$, $\widetilde{\mathsf{U}_{\mathbf{R}}} \coloneqq \mathsf{U}_{\widetilde{R}}\colon \widetilde{R} \nrightarrow \widetilde{R}$.
   The horizontal composition of 2-cells

$$
\begin{array}{ccccc}
\mathbf{R} & \xrightarrow{\ \mathbf{M}\ } & \mathbf{S} & \xrightarrow{\ \mathbf{N}\ } & \mathbf{T} \\
{\scriptstyle \mathbf{F}} \downarrow & \Downarrow \boldsymbol{\theta} & \downarrow {\scriptstyle \mathbf{G}} \quad \Downarrow \boldsymbol{\phi} & & \downarrow {\scriptstyle \mathbf{H}} \\
\mathbf{R'} & \xrightarrow[\ \mathbf{M'}\ ]{} & \mathbf{S'} & \xrightarrow[\ \mathbf{N'}\ ]{} & \mathbf{T'}
\end{array} \tag{41}
$$

is defined by the composition

$$
\begin{array}{ccccc}
\widetilde{R}^{\mathrm{op}} & \xrightarrow{\ \ M\ \ } & \mathbf{S}\text{-}\mathbf{Inst} & \xrightarrow{\ \ \Lambda_N\ \ } & \mathbf{T}\text{-}\mathbf{Inst} \\
{\scriptstyle \widetilde{F}^{\mathrm{op}}} \downarrow & \Downarrow \theta \quad {}^{\Delta_G}\!\nearrow\ \Downarrow\ \searrow^{\Sigma_G} & \Downarrow \Lambda_\phi & & \uparrow {\scriptstyle \Delta_H} \\
\widetilde{R'}^{\,\mathrm{op}} & \xrightarrow[\ \ M'\ \ ]{} \mathbf{S'}\text{-}\mathbf{Inst} & = & \mathbf{S'}\text{-}\mathbf{Inst} \xrightarrow[\ \ \Lambda_{N'}\ \ ]{} & \mathbf{T'}\text{-}\mathbf{Inst}
\end{array}
$$

where the middle triangle is the counit of the $\Sigma_G \dashv \Delta_G$ *adjunction. Vertical composition of 2-cells works in the evident way.*

   The data above satisfy the axioms of a double category as in Definition 2.8, with vertical category $\mathbb{D}\mathrm{ata}_0 = \mathbf{Schema}$ and horizontal $\mathcal{H}(\mathbb{D}\mathrm{ata})(\mathbf{R}, \mathbf{S}) = {}_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}$.

8.14. PROPOSITION. *The double category* $\mathbb{D}\mathrm{ata}$ *is an equipment.*

PROOF. It is clear from the definition of 2-cells in $\mathbb{D}\mathrm{ata}$ that given a niche

$$
\begin{array}{ccc}
\mathbf{R} & & \mathbf{S} \\
{\scriptstyle \mathbf{F}} \downarrow & & \downarrow {\scriptstyle \mathbf{G}} \\
\mathbf{R'} & \xrightarrow[\ \mathbf{N}\ ]{} & \mathbf{S'}
\end{array}
$$

there is a cartesian filler with the bimodule ${}_{\mathbf{F}}\mathbf{N}_{\mathbf{G}}$ from Section 8.12 on top. $\blacksquare$

8.15. REMARK. We deduce that the companion and conjoint of a schema mapping $\mathbf{F}\colon \mathbf{R} \to \mathbf{S}$ are the bimodules given by the following formulas:

$$\widehat{F} = \mathbf{y} \circ \widetilde{F}^{\mathrm{op}} \ : \ \widetilde{R}^{\mathrm{op}} \to \widetilde{S}^{\mathrm{op}} \to \mathbf{S\text{-}Inst}$$

$$\widecheck{F} = \Delta_F \circ \mathbf{y} \ : \ \widetilde{S}^{\mathrm{op}} \to \mathbf{S\text{-}Inst} \to \mathbf{R\text{-}Inst}$$

$$(42)$$

These bimodules turn out to be equivalent, via Theorem 8.10, to the companion and conjoint of the induced $\widetilde{F}\colon \widetilde{R} \to \widetilde{S}$ in the equipment $\mathbb{P}\mathbf{rof}$ (3). Moreover, due to Remark 6.9, $\widehat{F} \odot N \odot \widecheck{G}$ in $\mathbb{D}\mathbf{ata}$ coincides with $\widehat{\widetilde{F}} \odot \widetilde{N} \odot \widecheck{\widetilde{G}}$ in $\mathbb{P}\mathbf{rof}$, even though the horizontal compositions differ. This will be put into a larger context in Remark 8.32.

Recall from (37) the definition of Γ, which is right adjoint to Λ by Proposition 8.6.

8.16. PROPOSITION. *The equipment* $\mathbb{D}\mathbf{ata}$ *is right closed, with* $\mathbf{N} \rhd \mathbf{P} := \Gamma_N \circ P$.

PROOF. Let $\mathbf{M}\colon \mathbf{R} \nrightarrow \mathbf{S}$, $\mathbf{N}\colon \mathbf{S} \nrightarrow \mathbf{T}$ and $\mathbf{P}\colon \mathbf{R} \nrightarrow \mathbf{T}$. By Definition 2.9, it is enough to establish a natural bijection $_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{T}}(\mathbf{M} \odot \mathbf{N}, \mathbf{P}) \cong {}_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}(\mathbf{M}, \mathbf{N} \rhd \mathbf{P})$. This follows directly from the $\Lambda_N \dashv \Gamma_N$ adjunction:

$$\begin{aligned} _{\mathbf{R}}\mathrm{Bimod}_{\mathbf{T}}(\mathbf{M} \odot \mathbf{N}, \mathbf{P}) &= \mathbf{T\text{-}Inst}^{\widetilde{R}^{\mathrm{op}}}(\Lambda_N \circ M, P) \\ &\cong \mathbf{S\text{-}Inst}^{\widetilde{R}^{\mathrm{op}}}(M, \Gamma_N \circ P) \\ &= {}_{\mathbf{R}}\mathrm{Bimod}_{\mathbf{S}}(\mathbf{M}, \mathbf{N} \rhd \mathbf{P}), \end{aligned}$$

completing the proof. ∎

In the following, $\mathcal{L}\mathbf{Adj}_{\mathbf{Type}} \subseteq \mathbf{Type}^{\mathrm{op}}/\mathcal{C}\mathbf{at}$ and $\mathcal{R}\mathbf{Adj}_{\mathbf{Type}} \subseteq \mathcal{C}\mathbf{at}/\mathbf{Type\text{-}Alg}$ are the obvious sub-2-categories of the (co)slices described in Section 8.9.

8.17. PROPOSITION. *There is a commutative diagram of pseudofunctors and bicategories, each of which is a local equivalence:*

$$\begin{array}{ccc} & \xrightarrow{\ \Lambda_-\ } & \mathcal{L}\mathbf{Adj}_{\mathbf{Type}} \\ \mathcal{H}(\mathbb{D}\mathbf{ata}) & & \downarrow{\scriptstyle \simeq} \\ & \xrightarrow[\ \Gamma_-\ ]{} & \mathcal{R}\mathbf{Adj}_{\mathbf{Type}}^{\mathrm{op}} \end{array}$$

PROOF. On objects, Λ_- maps a schema $\mathbf{S}$ to the functor $\mathbf{y}\colon \mathbf{Type}^{\mathrm{op}} \to \mathbf{S\text{-}Inst}$; on bimodules and 2-cells, it is was described in Theorem 8.10. Then for any bimodules $\mathbf{M}\colon \mathbf{R} \nrightarrow \mathbf{S}$ and $\mathbf{N}\colon \mathbf{S} \nrightarrow \mathbf{T}$, we have $\Lambda_{M \odot N} := \Lambda_{\Lambda_N \circ M} \cong \Lambda_N \circ \Lambda_M$ and $\Lambda_{\mathbf{U_S}} := \Lambda_{\mathbf{y}} \cong \mathrm{id}_{\mathbf{S\text{-}Inst}}$ by Lemma 8.7.

By checking that the coherence axioms are satisfied, this establishes that Λ_- is a pseudofunctor. The result follows easily from there. ∎

The following lemma establishes a certain relationship between Λ, Γ and the data migration functors of Section 7. Recall that every schema mapping $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ induces a triple adjunction as on the left below, and that every bimodule $\mathbf{M} \colon \mathbf{S} \nrightarrow \mathbf{T}$ induces an adjunction as on the right:

$$
\mathbf{T}\text{-}\mathbf{Inst} \quad
\overset{\Sigma_F}{\underset{\Pi_F}{
\begin{array}{c}
\xleftarrow{\hspace{1.5cm}} \\[-4pt]
\perp \\[-4pt]
\xrightarrow{\ \Delta_F\ } \\[-4pt]
\perp \\[-4pt]
\xleftarrow{\hspace{1.5cm}}
\end{array}}}
\quad \mathbf{S}\text{-}\mathbf{Inst}
\qquad\qquad
\mathbf{T}\text{-}\mathbf{Inst} \quad
\overset{\Lambda_M}{\underset{\Gamma_M}{
\begin{array}{c}
\xleftarrow{\hspace{1cm}} \\[-4pt]
\perp \\[-4pt]
\xrightarrow{\hspace{1cm}}
\end{array}}}
\quad \mathbf{S}\text{-}\mathbf{Inst}
$$

Recall also that every schema mapping $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ induces a pair of bimodules $\widehat{F} \colon \mathbf{S} \nrightarrow \mathbf{T}$ and $\breve{F} \colon \mathbf{T} \nrightarrow \mathbf{S}$.

8.18. LEMMA. *For any schema mapping $\mathbf{F}$, we have the following isomorphisms and adjunctions:*

$$
(\Sigma_F \cong \Lambda_{\widehat{F}}) \ \dashv \ (\Delta_F \cong \Lambda_{\breve{F}} \cong \Gamma_{\widehat{F}}) \ \dashv \ (\Pi_F \cong \Gamma_{\breve{F}})
$$

PROOF. The adjunctions are given in Propositions 7.3 and 7.4, so we provide the isomorphisms. The companion and conjoint of schema mappings are given in (42).

For $\mathbf{F} \colon \mathbf{S} \to \mathbf{T}$ and an instance $\mathbf{I} \in \mathbf{S}\text{-}\mathbf{Inst}$, we have an isomorphism

$$
\Lambda_{\widehat{F}}(I) = \int^{s \in \widetilde{S}} I(s) \cdot \widehat{F}(s) \cong \int^{s \in \widetilde{S}} I(s) \cdot \widetilde{T}(\widetilde{F}s, -) \cong \Sigma_F(I)
$$

by (35), Remark 8.15 and (31). For any $s \in \widetilde{S}$ we also have an isomorphism

$$
\Gamma_{\breve{F}} I(s) = \mathbf{S}\text{-}\mathbf{Inst}(\breve{F}(s), I) \cong \mathbf{S}\text{-}\mathbf{Inst}(\Delta_F(\mathbf{y}s), I) \cong \Pi_F I(s)
$$

by (37), Remark 8.15, and Proposition 7.3. The remaining isomorphisms (for Δ) follow by Proposition 8.6. ∎

8.19. DECOMPOSING BIMODULES. Let $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ be a bimodule. Since $\mathbb{P}\mathbf{rof}$ has extensive collages by Example 2.27, the respective profunctor $\widetilde{M} \colon \widetilde{R} \nrightarrow \widetilde{S}$ determines an (R_o, S_o)-simplex in the sense of Definition 2.21: four profunctors $M_{ee} \colon R_e \nrightarrow S_e$, $M_{et} \colon R_e \nrightarrow \mathbf{Type}$, $M_{te} \colon \mathbf{Type} \nrightarrow S_e$, and $M_{tt} \colon \mathbf{Type} \nrightarrow \mathbf{Type}$, obtained via the restriction of M along the obvious inclusions, together with four 2-cells M_{e*}, M_{t*}, M_{*e}, and M_{*t}.

The conditions of Remark 8.4 force M_{te} to be the initial profunctor (i.e. the constant functor $\mathbf{Type}^{\mathrm{op}} \times S_e \to \mathbf{Set}$ with value the empty set), M_{tt} to be the unit profunctor (i.e. the hom functor $\mathbf{Type}^{\mathrm{op}} \times \mathbf{Type} \to \mathbf{Set}$) and M_{et} to be algebraic. Because M_{te} is initial, and because tensor product of profunctors preserves colimits, the 2-cells M_{t*} and M_{*e} are unique, and hence don't need to be specified. Thus we have proven the following proposition, in which we let $M_e \coloneqq M_{ee}$, $M_t \coloneqq M_{et}$, $M_o \coloneqq M_{e*}$, and $M_r \coloneqq M_{*t}$.[5]

---

[5] The mnemonic for M_r comes from its role as "return clause" in queries; see Section 9.4.

8.20. PROPOSITION. *A bimodule* $\mathbf{M} \colon \mathbf{R} \twoheadrightarrow \mathbf{S}$ *is equivalent to a tuple* $(M_{\mathrm{e}}, M_{\mathrm{t}}, M_{\mathrm{o}}, M_{\mathrm{r}})$, *where* $M_{\mathrm{e}} \colon R_{\mathrm{e}} \twoheadrightarrow S_{\mathrm{e}}$ *is a profunctor,* $M_{\mathrm{t}} \colon R_{\mathrm{e}} \twoheadrightarrow \mathbf{Type}$ *is an algebraic profunctor, and* M_{o} *and* M_{r} *are profunctor morphisms*

$$\begin{array}{c}\xrightarrow{M_{\mathrm{e}}}\ S_{\mathrm{e}}\ \xrightarrow{S_{\mathrm{o}}}\\ \Downarrow M_{\mathrm{o}}\\ R_{\mathrm{e}} \xrightarrow[M_{\mathrm{t}}]{} \mathbf{Type}\\ \Uparrow M_{\mathrm{r}}\\ R_{\mathrm{o}}\end{array} \tag{43}$$

8.21. EXAMPLE. For any schema $\mathbf{R}$, the unit bimodule $\mathsf{U}_{\mathbf{R}} \colon \mathbf{R} \twoheadrightarrow \mathbf{R}$ is given by

$$\begin{array}{c}\xrightarrow{R_{\mathrm{e}}}\ R_{\mathrm{e}}\ \xrightarrow{R_{\mathrm{o}}}\\ \mathrm{id}\\ R_{\mathrm{e}} \xrightarrow[R_{\mathrm{o}}]{} \mathbf{Type}\\ \mathrm{id}\\ R_{\mathrm{o}}\end{array} \tag{44}$$

and the companion/conjoint of a schema mapping $\mathbf{F} = (F_{\mathrm{e}}, F_{\mathrm{o}}) \colon \mathbf{R} \to \mathbf{S}$ decompose as

$$\begin{array}{c}\xrightarrow{\widehat{F}_{\mathrm{e}}}\ S_{\mathrm{e}}\ \xrightarrow{S_{\mathrm{o}}}\\ \Downarrow \mathrm{id}\\ R_{\mathrm{e}} \xrightarrow[\widehat{F}_{\mathrm{e}} \odot S_{\mathrm{o}}]{} \mathbf{Type}\\ \Uparrow \widehat{F}_{\mathrm{o}}\\ R_{\mathrm{o}}\end{array} \qquad \begin{array}{c}\xrightarrow{\widecheck{F}_{\mathrm{e}}}\ R_{\mathrm{e}}\ \xrightarrow{R_{\mathrm{o}}}\\ \Downarrow \widecheck{F}_{\mathrm{o}}\\ S_{\mathrm{e}} \xrightarrow[S_{\mathrm{o}}]{} \mathbf{Type}\\ \Uparrow \mathrm{id}\\ S_{\mathrm{o}}\end{array}$$

where 2-cells $\widehat{F}_{\mathrm{o}}, \widecheck{F}_{\mathrm{o}}$ are as in Section 2.11 for $\mathbb{P}\mathbf{rof}$. This is 'component-wise' Remark 8.15.

The equivalence ${}_F\mathrm{Res}_G$ (Section 2.22) for the extensive collages equipment $\mathbb{P}\mathbf{rof}$, which on objects resulted in Proposition 8.20, also gives an equivalent expression of a 2-cell $\boldsymbol{\theta}$ in $\mathbb{D}\mathbf{ata}$, viewed as $M \to \widehat{\widetilde{F}} \odot N \odot \widecheck{\widetilde{G}}$ inside $\mathcal{H}(\mathbb{P}\mathbf{rof})(\widetilde{S}, \widetilde{R})$ (see Remark 8.15).

8.22. PROPOSITION. *A 2-cell* $\boldsymbol{\theta}$ *in* $\mathbb{D}\mathbf{ata}$ (39) *is equivalently a pair of profunctor morphisms*

$$\begin{array}{ccc} R_{\mathrm{e}} & \xrightarrow{M_{\mathrm{e}}} & S_{\mathrm{e}}\\ F_{\mathrm{e}} \downarrow & \Downarrow \theta_{\mathrm{e}} & \downarrow G_{\mathrm{e}}\\ R'_{\mathrm{e}} & \xrightarrow[M'_{\mathrm{e}}]{} & S'_{\mathrm{e}} \end{array} \qquad \begin{array}{ccc} R_{\mathrm{e}} & \xrightarrow{M_{\mathrm{t}}} & \mathbf{Type}\\ F_{\mathrm{e}} \downarrow & \Downarrow \theta_{\mathrm{t}} & \|\\ R'_{\mathrm{e}} & \xrightarrow[M'_{\mathrm{t}}]{} & \mathbf{Type} \end{array}$$

satisfying the equations

$$(45)$$

8.23. COROLLARY. *A 2-cell $\boldsymbol{\theta}$ in $\mathbb{D}$ata is cartesian if and only if the 2-cells θ_e and θ_t from Proposition 8.22 are cartesian in $\mathbb{P}$rof.*

Because it will be convenient later, we now present yet another equivalent representation of bimodules, which is in some sense intermediate between Definition 8.3 and the completely decomposed representation of Proposition 8.20. Recall from Example 7.2 that for any **S**-instance **I**, the underlying **Type**-algebra is given by $I_t = \Delta_{!_S}(\mathbf{I})$, where $!_S \colon \emptyset \to \mathbf{S}$ is the unique map.

8.24. PROPOSITION. *A bimodule $\mathbf{M} \colon \mathbf{R} \nrightarrow \mathbf{S}$ is equivalent to a functor $M_o \colon R_e^{op} \to$* **S-Inst** *together with a natural transformation*

PROOF. The functor $M^{op} \colon \widetilde{R} \to$ **S-Inst**^{op}, opposite to the one given in Definition 8.3, can equivalently be defined — using the universal property (11) of collages in $\mathbb{P}$rof — as a functor $M_o^{op} \colon R_e \to$ **S-Inst**^{op}, along with a natural transformation

since by definition, types are mapped to representables. This natural family of functions $R_o(r, \tau) \to$ **S-Inst** $(\mathbf{y}(\tau), M_o(r))$ equivalently define M_r as a natural transformation $R_o \Rightarrow \Delta_{!_S} \circ M_o$ by Yoneda: **S-Inst**$(\mathbf{y}(\tau), M_o(r)) \cong M_o(r)(\tau)$. ∎

8.25. INSTANCES IN TERMS OF BIMODULES. The category of instances on a schema $\mathbf{S}$ can be viewed entirely in terms of bimodules. Indeed, if $\mathbf{U} = (\{*\}, \kappa)$ is the unit schema from Example 5.4, then we have an isomorphism of categories

$$\mathcal{H}(\mathbb{Data})(\mathbf{U}, \mathbf{S}) \cong \mathbf{S}\text{-Inst}.$$

This follows by comparing their decomposed forms — see Section 6.19 and Propositions 8.20 and 8.22 — and using the fact that $\kappa\colon \{*\} \twoheadrightarrow \mathbf{Type}$ is the initial $\mathbf{Type}$-algebra.

It also follows that $\Lambda_N(-)$ is simply given by bimodule composition. Indeed, by (40), for any bimodule $\mathbf{N}\colon \mathbf{S} \twoheadrightarrow \mathbf{T}$ and $\mathbf{S}$-instance $\mathbf{J}$, considered as a $(\mathbf{U}, \mathbf{S})$-bimodule, one has

$$\Lambda_N(\mathbf{J}) \cong \mathbf{J} \odot \mathbf{N}. \tag{46}$$

Similarly, for any $\mathbf{T}$-instance I,

$$\Gamma_N(\mathbf{I}) \cong \mathbf{J} \triangleright \mathbf{N}.$$

8.26. DATA MIGRATION FUNCTORS IN TERMS OF BIMODULES. We can also recover the fundamental data migration functors from the structure of $\mathbb{Data}$, using Lemma 8.18 and (46). That is, if we consider instances as bimodules $\mathbf{I}\colon \mathbf{U} \twoheadrightarrow \mathbf{R}$ and $\mathbf{J}\colon \mathbf{U} \twoheadrightarrow \mathbf{S}$, then composing and exponentiating them with companions and conjoints of $\mathbf{F}\colon \mathbf{R} \to \mathbf{S}$ is equivalent to applying Σ, Δ, Π:

$$\Sigma_F(I) \cong \mathbf{I} \odot \widehat{\mathbf{F}}, \qquad \Delta_F(J) \cong \mathbf{J} \odot \widecheck{\mathbf{F}} \cong \widehat{\mathbf{F}} \triangleright \mathbf{J}, \qquad \Pi_F(I) \cong \widecheck{\mathbf{F}} \triangleright \mathbf{I}$$

8.27. COLLAGES IN $\mathbb{Data}$. We now consider collages (see Definition 2.16) in the proarrow equipment $\mathbb{Data}$. Using Proposition 8.20 and the fact that $\mathbf{Prof}$ has extensive collages (Example 2.27), we can fully express a collage in $\mathbb{Data}$ in terms of profunctor collages.

Let $\mathbf{M} = (M_e, M_t, M_o, M_r)\colon \mathbf{R} \twoheadrightarrow \mathbf{S}$ be a bimodule as in (43). Its collage will be a schema $\mathbf{Col}(\mathbf{M})$, together with two schema mappings $\mathbf{R} \to \mathbf{Col}(\mathbf{M}) \leftarrow \mathbf{S}$ and a universal 2-cell $\boldsymbol{\mu}\colon \mathbf{M} \Rightarrow \mathsf{U}_{\mathbf{Col}(\mathbf{M})}$ in $\mathbb{Data}$. We begin by describing $\mathbf{Col}(\mathbf{M})$.

8.28. THE SCHEMA OF A BIMODULE COLLAGE. The entity category of the collage $\mathbf{Col}(\mathbf{M})$ is the collage of the profunctor $M_e\colon R_e \twoheadrightarrow S_e$

$$\mathrm{Col}(M)_e \coloneqq \widetilde{M_e},$$

and the observables profunctor $\mathrm{Col}(M)_o\colon \widetilde{M_e} \twoheadrightarrow \mathbf{Type}$ is the one uniquely corresponding, via the universal property of the lax colimit $\widetilde{M_e}$ (dual of Remark 2.32), to the cocone
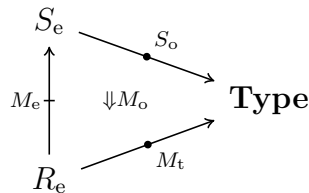
In simple words, the functor $\mathrm{Col}(M)_o\colon \widetilde{M_e}^{\mathrm{op}} \times \mathbf{Type} \to \mathbf{Set}$ is given by M_t on the R_e-side of $\widetilde{M_e}$, by S_o on the S_e-side of $\widetilde{M_e}$, and by M_o on the morphisms in between. The profunctor $\mathrm{Col}(M)_o$ is algebraic, because M_t and S_o are.

8.29. The schema mappings of a bimodule collage. We now define the collage inclusions $\mathbf{i_R} \colon \mathbf{R} \to \mathbf{Col(M)} \leftarrow \mathbf{S} \colon \mathbf{i_S}$. They are schema mappings as in Definition 5.10, thus each consists of a functor between entity categories and a 2-cell in $\mathbb{P}\mathbf{rof}$. The functors between entity categories are the collage inclusions from $\mathbb{P}\mathbf{rof}$ (see Example 2.19):

$$(i_\mathbf{R})_\mathrm{e} \coloneqq i_{R_\mathrm{e}} \colon R_\mathrm{e} \to \widetilde{M}_\mathrm{e} \qquad \text{and} \qquad (i_\mathbf{S})_\mathrm{e} \coloneqq i_{S_\mathrm{e}} \colon S_\mathrm{e} \to \widetilde{M}_\mathrm{e}.$$

The 2-cells $(i_\mathbf{R})_\mathrm{o}$ and $(i_\mathbf{S})_\mathrm{o}$ in $\mathbb{P}\mathbf{rof}$ are defined respectively as follows:

(47)

The fact that the indicated 2-cells are cartesian follows by definition of $\mathbf{Col(M)}$ as a lax colimit; see Proposition 2.30.

8.30. The 2-cell of a bimodule collage. We now define the 2-cell $\boldsymbol{\mu}$ in $\mathbb{D}\mathbf{ata}$

(48)

in its decomposed form (see Proposition 8.22) to be the pair $\boldsymbol{\mu} = (\mu_\mathrm{e}, \mu_\mathrm{t})$

where μ_e is the universal 2-cell for $\widetilde{M}_\mathrm{e}$ in $\mathbb{P}\mathbf{rof}$ (see Example 2.19) and μ_t is the (cartesian) square shown to the left in (47). The components μ_e and μ_t satisfy the equations (45) by the unit bimodule decomposition (44) and the universal property of lax colimit (20).

8.31. Proposition. *The equipment* $\mathbb{D}\mathbf{ata}$ *has normal collages.*

Note that $\mathbb{D}\mathbf{ata}$ does not have extensive collages. In particular, i_R is not in general fully faithful.

PROOF. We must first verify that $\mathbb{Data}$ has collages, i.e. that the 2-cell $\boldsymbol{\mu}$ defined in (48) has the required universal property (11). Suppose that $\mathbf{X}$ is a schema and that $\boldsymbol{\phi}\colon \mathbf{M} \Rightarrow \mathsf{U}_{\mathbf{X}}$ is a 2-cell from $\mathbf{M}$ to the unit bimodule. We must show that $\boldsymbol{\phi}$ factors uniquely through $\boldsymbol{\mu}$:

$$
\begin{array}{ccc}
\begin{array}{ccc}
\mathbf{R} & \xrightarrow{\ \mathbf{M}\ } & \mathbf{S} \\
{\scriptstyle \mathbf{F}}\downarrow & \Downarrow\phi & \downarrow{\scriptstyle \mathbf{G}} \\
\mathbf{X} & \xrightarrow[\mathbf{X}]{} & \mathbf{X}
\end{array}
& = &
\begin{array}{ccc}
\mathbf{R} & \xrightarrow{\ \mathbf{M}\ } & \mathbf{S} \\
{\scriptstyle i_{\mathbf{R}}}\downarrow & \Downarrow\mu & \downarrow{\scriptstyle i_{\mathbf{S}}} \\
\mathbf{Col(M)} & \xrightarrow[\mathbf{Col(M)}]{} & \mathbf{Col(M)} \\
{\scriptstyle \bar{\phi}}\downarrow & \Downarrow\bar{\phi} & \downarrow{\scriptstyle \bar{\phi}} \\
\mathbf{X} & \xrightarrow[\mathbf{X}]{} & \mathbf{X}
\end{array}
\end{array}
$$

We work with components, writing $\boldsymbol{\phi} = (\phi_{\mathrm{e}}, \phi_{\mathrm{t}})$ as in Proposition 8.22. Firstly, since μ_{e} is the universal 2-cell for a collage in $\mathbb{Prof}$, we have that $\phi_{\mathrm{e}} = \mathsf{U}_{\bar{\phi}_{\mathrm{e}}} \circ \mu_{\mathrm{e}}$ for a unique functor $\bar{\phi}_{\mathrm{e}}\colon \widetilde{M_{\mathrm{e}}} \to X_{\mathrm{e}}$. Also, $\phi_{\mathrm{t}} = \bar{\phi}_{\mathrm{t}} \circ \mu_{\mathrm{t}}$ as in

$$
\begin{array}{ccc}
\begin{array}{ccc}
R_{\mathrm{e}} & \xrightarrow{\ M_{\mathrm{t}}\ } & \mathbf{Type} \\
{\scriptstyle F_{\mathrm{e}}}\downarrow & \Downarrow\phi_{\mathrm{t}} & \| \\
X_{\mathrm{e}} & \xrightarrow[X_{\mathrm{o}}]{} & \mathbf{Type}
\end{array}
& = &
\begin{array}{ccc}
R_{\mathrm{e}} & \xrightarrow{\ M_{\mathrm{t}}\ } & \mathbf{Type} \\
{\scriptstyle i_{R_{\mathrm{e}}}}\downarrow & \Downarrow\mu_{\mathrm{t}} & \| \\
\widetilde{M_{\mathrm{e}}} & \xrightarrow[\mathrm{Col}(M)_{\mathrm{o}}]{} & \mathbf{Type} \\
{\scriptstyle \bar{\phi}_{\mathrm{e}}}\downarrow & \Downarrow\bar{\phi}_{\mathrm{t}} & \| \\
X_{\mathrm{e}} & \xrightarrow[X_{\mathrm{o}}]{} & \mathbf{Type}
\end{array}
\end{array}
$$

for a unique 2-cell $\bar{\phi}_{\mathrm{t}}$, obtained via the 2-dimensional universal property of the lax colimit $\widetilde{M_{\mathrm{e}}}$ (Proposition 2.30). This profunctor morphism $\bar{\phi}_{\mathrm{t}}\colon \mathrm{Col}(M)_{\mathrm{o}} \Rightarrow \widehat{\bar{\phi}}_{\mathrm{e}} \odot X_{\mathrm{o}}$ is concretely defined, omitting the details, by natural components

$$
(\bar{\phi}_{\mathrm{t}})_{r\tau} = \mathrm{Col}(M)_{\mathrm{o}}(r, \tau) \rightsquigarrow M_{\mathrm{t}}(r, \tau) \xrightarrow{(\phi_{\mathrm{t}})_{r\tau}} X_{\mathrm{o}}(F_{\mathrm{e}}r, \tau) = X_{\mathrm{o}}(\bar{\phi}_{\mathrm{e}}r, \tau)
$$

$$
(\bar{\phi}_{\mathrm{t}})_{s\tau} = \mathrm{Col}(M)_{\mathrm{o}}(s, \tau) \rightsquigarrow S_{\mathrm{o}}(s, \tau) \xrightarrow{(G_{\mathrm{o}})_{s\tau}} X_{\mathrm{o}}(G_{\mathrm{e}}s, \tau) = X_{\mathrm{o}}(\bar{\phi}_{\mathrm{e}}s, \tau)
$$

for $r \in R_{\mathrm{e}}$, $s \in S_{\mathrm{e}}$ and $\tau \in \mathbf{Type}$. Defining $\bar{\boldsymbol{\phi}}$ to be the pair $(\mathsf{U}_{\bar{\phi}_{\mathrm{e}}}, \bar{\phi}_{\mathrm{t}})\colon \mathbf{Col(M)} \to \mathbf{X}$, we have $\boldsymbol{\phi} = \mathsf{U}_{\bar{\phi}} \circ \boldsymbol{\mu}$ as desired.

Moreover, collages in $\mathbb{Data}$ are normal, as in Definition 2.18, since the 2-cell $\boldsymbol{\mu}$ constructed in Section 8.30 is cartesian: by Corollary 8.23, it is enough that μ_{e} and μ_{t} are cartesian liftings in $\mathbb{Prof}$. ∎

8.32. REMARK. Although we will not use this fact, we note that the collage correspondences from Remarks 5.3, 5.12 and 8.4 provide a *lax double functor* (see e.g. [GP04])

$$
\widetilde{(-)}\colon \mathbb{Data} \to \mathbb{Prof}.
$$

This functor is only lax, because for bimodules $\mathbf{R} \xrightarrow{\mathbf{M}} \mathbf{S} \xrightarrow{\mathbf{N}} \mathbf{T}$ in $\mathbb{D}\mathbf{ata}$, the natural map $\widetilde{M} \odot \widetilde{N} \to \widetilde{M \odot N}$ in $\mathbb{P}\mathbf{rof}$ given by the unique transformation in $[\widetilde{T}, \mathbf{Set}]$

$$\int^{s \in \widetilde{S}} \widetilde{M}(r, s) \times \widetilde{N}(s, -) \Rightarrow \int^{s \in \widetilde{S}} M(r)(s) \cdot N(s)$$

between the pointwise colimit (4) and the $\mathbf{T}$-$\mathbf{Inst}$-one (40), is not an isomorphism (Remark 6.9). Since lax double functors between equipments automatically preserve cartesian liftings (see [Shu08, Prop. 6.4]), this fact also explains Remark 8.15.

8.33. BIMODULES IN TERMS OF DATA MIGRATION. We will now see that any bimodule, considered as an adjoint functor on instance categories via Theorem 8.10, is equivalent to a composite of data migration functors.

8.34. COROLLARY. *Let* $\mathbf{M} \colon \mathbf{R} \rightarrowtail \mathbf{S}$ *be a bimodule, and let* $\mathbf{i_R} \colon \mathbf{R} \to \mathbf{Col}(\mathbf{M})$ *and* $\mathbf{i_S} \colon \mathbf{S} \to \mathbf{Col}(\mathbf{M})$ *be the collage inclusions. We have isomorphisms*

$$\Lambda_M \cong \Delta_{i_\mathbf{S}} \circ \Sigma_{i_\mathbf{R}} \quad \text{and} \quad \Gamma_M \cong \Delta_{i_\mathbf{R}} \circ \Pi_{i_\mathbf{S}}$$

PROOF. Since $\mathbb{D}\mathbf{ata}$ has normal collages, (48) is a cartesian 2-cell, hence $\mathbf{M} \cong \widehat{i_\mathbf{R}} \odot \mathbf{U_{Col(M)}} \odot \widecheck{i_\mathbf{S}}$. Therefore, by Lemma 8.18 and Proposition 8.17, we have

$$\Lambda_M \cong \Lambda_{\widehat{i_\mathbf{R}} \odot \mathbf{U_{Col(M)}} \odot \widecheck{i_\mathbf{S}}} \cong \Lambda_{\widecheck{i_\mathbf{S}}} \circ \Lambda_{\widehat{i_\mathbf{R}}} \cong \Delta_{i_\mathbf{S}} \circ \Sigma_{i_\mathbf{R}}$$

and dually for the right adjoint Γ_M. ∎

8.35. COROLLARY. *Suppose that* $\mathbf{R}$ *and* $\mathbf{S}$ *be schemas. Let* $\mathscr{L} \subseteq \mathcal{L}\mathbf{Adj_{Type}}(\mathbf{R}\text{-}\mathbf{Inst}, \mathbf{S}\text{-}\mathbf{Inst})$ *[resp. let* $\mathscr{R} \subseteq \mathcal{R}\mathbf{Adj_{Type}}(\mathbf{S}\text{-}\mathbf{Inst}, \mathbf{R}\text{-}\mathbf{Inst})$*] denote the full subcategory spanned by functors of the form* $\Delta_G \circ \Sigma_F$ *[resp. of the form* $\Pi_G \circ \Delta_F$*]. This inclusion is an equivalence of categories.*

PROOF. The inclusion functor is fully faithful by definition and essentially surjective by Corollary 8.34 and Theorem 8.10. ∎

8.36. REMARK. Corollary 8.35 says that every right adjoint between instance categories is naturally isomorphic to a right pushforward followed by a pullback, $\Delta \circ \Pi$. While we do not discuss the details here, there is a similar characterization of *parametric right adjoints* (a.k.a. *local right adjoints*) between instance categories.

Recall that a functor $F \colon \mathscr{C} \to \mathscr{D}$ is a parametric right adjoint if, for each object $c \in \mathscr{C}$, the slice $F/c \colon \mathscr{C}/c \to \mathscr{D}/(Fc)$ is a right adjoint, see e.g. [Web07]. In our setting, one can show that every parametric right adjoint between instance categories is naturally isomorphic to a functor of the form $\Sigma_{\mathrm{dopf}} \circ \Delta \circ \Pi$, where the left pushfoward is along a discrete op-fibration, as discussed in Section 7.11. This generalizes the analogous fact for parametric right adjoints between presheaf categories, as shown in [Web07, Remark 2.12].

8.37. BIMODULES PRESENTATION. We conclude Section 8 by discussing presentations of bimodules, which work very similarly to presentations of profunctors (Definition 4.45). Recall also the definition of schema presentations, Definition 5.6. Suppose that **Type** $\cong$ $\mathbf{Cxt}_\Sigma/E_\Sigma$ has algebraic signature Σ;[6] see Definition 4.18.

8.38. DEFINITION. *Let* $\mathbf{R}$ *and* $\mathbf{R}'$ *be schemas given respectively by presentations* $(\Xi, E_\mathrm{e}, E_\mathrm{o})$ *and* $(\Xi', E_\mathrm{e}', E_\mathrm{o}')$. *These present entity category* $R_\mathrm{e} \cong \mathrm{Fr}(G_\Xi)/E_\mathrm{e}$ *and observables profunctor* $R_\mathrm{o} \cong \kappa[\Upsilon_\Xi]/E_\mathrm{o}$, *and similarly for* $\mathbf{R}'$.

A bimodule signature $\Omega = (\Omega_\mathrm{e}, \Omega_\mathrm{o})$ *from* Ξ *to* Ξ' *is a pair where* Ω_e *is a profunctor signature from* G_Ξ *to* $G_{\Xi'}$, *and* Ω_o *is a profunctor signature from* G_Ξ *to* Σ.

A bimodule signature has an associated algebraic signature $\widetilde{\Omega} = (S_{\widetilde{\Omega}}, \Phi_{\widetilde{\Omega}})$, *where*

$$S_{\widetilde{\Omega}} = (G_\Xi)_0 \sqcup (G_{\Xi'})_0 \sqcup S_\Sigma$$
$$\Phi_{\widetilde{\Omega}} = (G_\Xi)_1 \sqcup (G_{\Xi'})_1 \sqcup \Phi_\Sigma \sqcup \Upsilon_\Xi \sqcup \Upsilon_{\Xi'} \sqcup \Omega_\mathrm{e} \sqcup \Omega_\mathrm{o}.$$

Say that a set E_Ω *of equations over* $\widetilde{\Omega}$ *is a set of* bimodule equations *if for each equation* $\Gamma \vdash (t_1 = t_2) : s'$ *of* E_Ω, *the context is a singleton* $\Gamma = (x : s)$ *with* $s \in (G_\Xi)_0$ *and* $s' \in (G_{\Xi'})_0 \sqcup S_\Sigma$. *We can partition the set* $E_\Omega = (E_\Omega)_\mathrm{e} \sqcup (E_\Omega)_\mathrm{o}$, *where* $(E_\Omega)_\mathrm{e}$ *contains all equations where* $s' \in (G_{\Xi'})_0$, *and* $(E_\Omega)_\mathrm{o}$ *contains all equations where* $s' \in S_\Sigma$.

Given a pair (Ω, E_Ω), *consider the category* $\mathbf{Cxt}_{\widetilde{\Omega}}/E_{\widetilde{\Omega}}$, *where*

$$E_{\widetilde{\Omega}} = E_\mathrm{e} \cup E_\mathrm{o} \cup E_\mathrm{e}' \cup E_\mathrm{o}' \cup E_\Omega.$$

The bimodule $M = \kappa[\Omega]/E_\Omega$ *presented by* (Ω, E_Ω) *is defined as follows:*
- *for any objects* $r \in \widetilde{R}$ *and* $s \in \widetilde{R}'$, *the set* $M(r, r')$ *is defined to be the hom-set* $(\mathbf{Cxt}_{\widetilde{\Omega}}/E_{\widetilde{\Omega}})(r, s)$,
- *the functorial actions are given by substitution.*

8.39. REMARK. The presented bimodule $M = \kappa[\Omega]/E_\Omega$ may be easier to understand in terms of its collage. We will write $\mathrm{Col}(\Omega) = (G_{\mathrm{Col}(\Omega)}, \Upsilon_{\mathrm{Col}(\Omega)})$ for the following schema presentation:

$$(G_{\mathrm{Col}(\Omega)})_0 = (G_\Xi)_0 \sqcup (G_{\Xi'})_0$$
$$(G_{\mathrm{Col}(\Omega)})_1 = (G_\Xi)_1 \sqcup (G_{\Xi'})_1 \sqcup \Omega_\mathrm{e}$$
$$\Upsilon_{\mathrm{Col}(\Omega)} = \Upsilon_\Xi \sqcup \Upsilon_{\Xi'} \sqcup \Omega_\mathrm{o}.$$

It is easy to see that the algebraic signature $\widetilde{\mathrm{Col}(\Omega)}$ corresponding to the schema signature $\mathrm{Col}(\Omega)$ as in Definition 5.6 is precisely the same as the signature $\widetilde{\Omega}$ given above. Moreover, the collage $\mathbf{Col}(\mathbf{M})$ of the bimodule M is presented by $(\mathrm{Col}(\Omega), E_\mathrm{e} \cup E_\mathrm{e}' \cup (E_\Omega)_\mathrm{e}, E_\mathrm{o} \cup E_\mathrm{o}' \cup (E_\Omega)_\mathrm{o})$.

The inclusions $\mathbf{i_R}$ and $\mathbf{i_S}$ of the schemas $\mathbf{R}$ and $\mathbf{S}$ into the collage $\mathbf{Col}(\mathbf{M})$ are also easy to understand in terms of this presentation, as they are both inclusions on the level of generators and equations as well.

---

[6]Signatures Σ should not be confused with data migration functors Σ_-.

8.40. EXAMPLE. Let $\mathbf{F} = (F_e, F_o)\colon \mathbf{R} \to \mathbf{S}$ be a schema morphism. Both its companion $\widehat{\mathbf{F}}$ and its conjoint $\widecheck{\mathbf{F}}$ have very simple presentations.

The generators of $\widehat{\mathbf{F}}\colon \mathbf{R} \twoheadrightarrow \mathbf{S}$ are $\psi_r\colon r \to F_e(r)$ for each $r \in R_e$. For each edge $f\colon r \to r'$ in $\mathbf{R}$ there is an equation $x.f.\psi_{r'} = x.\psi_r.F_e(f)$, and for each attribute $\mathtt{att}\colon r \to \tau$ in $\mathbf{R}$ there is an equation $x.\mathtt{att} = x.\psi_r.F_o(\mathtt{att})$, both in context $(x : r)$.

The generators of $\widecheck{\mathbf{F}}\colon \mathbf{S} \twoheadrightarrow \mathbf{R}$ are $\phi_r\colon F_e(r) \to r$ for each $r \in R_e$. For each edge $f\colon r \to r'$ in $\mathbf{R}$ there is an equation $x.\phi_r.f = x.F_e(f).\phi_{r'}$, and for each attribute $\mathtt{att}\colon r \to \tau$ in $\mathbf{R}$ there is an equation $x.\phi_r.\mathtt{att} = x.F_o(\mathtt{att})$, both in context $(x : F_e(r))$.

9. Queries and uber-queries

In this section, we will employ many of the concepts and operations studied so far in order to describe the process of *querying* an algebraic database. We will also give examples that tie in with running examples from previous sections.

A query is a question asked of a database, such as "Tell me the set of employees whose manager is named Alice". Queries are often written using "Select-From-Where" — e.g. in the database query language SQL — or equivalently using "For-Where-Return" syntax. This syntax both poses the question and provides a table layout in which to record the results.

In our current setting, we will express a query on a given $\mathbf{S}$-instance $\mathbf{J}$, by constructing a new schema $\mathbf{R}$ and a bimodule $\mathbf{M}\colon \mathbf{R} \twoheadrightarrow \mathbf{S}$. Running the query will amount to applying the functor $\Gamma_M\colon \mathbf{S}\text{-}\mathbf{Inst} \to \mathbf{R}\text{-}\mathbf{Inst}$ from (37). Classically, a For-Where-Return query returns a single table (with no foreign keys), so the result schema $\mathbf{R}$ has a very specific form; namely, its entity side is the terminal category, $R_e = \{*\}$.

If we allow arbitrary $\mathbf{R}$ and arbitrary bimodules $\mathbf{R} \twoheadrightarrow \mathbf{S}$, the "query" Γ_M could be thought of as a method of migrating data from $\mathbf{S}$ to $\mathbf{R}$, but it could also be considered as a collection of queries and homomorphisms between them; we refer to such a setup as an *uber-query*. We will discuss this interpretation of bimodules in Section 9.8.

9.1. QUERIES. We begin by discussing the usual For-Where-Return queries and how they appear in our setup.

9.2. DEFINITION. *Let $\mathbf{S}$ be a schema given by a presentation (Ξ, E), see Definition 5.6. A* query *on $\mathbf{S}$ is a 4-tuple $Q = (Q_f, Q_w, Q_a, Q_r)$, where Q_f is a context over $\widetilde{\Xi}$, Q_w is a set of equations in Q_f, Q_a is a context over (the signature of) $\mathbf{Type}$, and $Q_r\colon Q_f \to Q_a$ is a context morphism over $\widetilde{\Xi}$.*

We will adopt the For-Where-Return notation for presenting the data of a query as defined in Definition 9.2, as follows:

$$
\begin{aligned}
\text{FOR:} &\quad Q_f \\
\text{WHERE:} &\quad Q_w \\
\text{RETURN:} &\quad Q_r
\end{aligned}
$$

This notation is sometimes called *flower syntax* (an acronym of For-Let-Where-Return) or *comprehension syntax* [AHV95].

9.3. EXAMPLE. Let **S** be the schema from Example 5.8. We give an example query Q on **S**:

$$\begin{aligned}
\text{FOR:} \quad & e : \texttt{Emp}, \, d : \texttt{Dept} \\
\text{WHERE:} \quad & e.\textsf{wrk}.\textsf{name} = \text{Admin}, \\
& (e.\textsf{sal} \leq d.\textsf{sec}.\textsf{sal}) = \top \\
\text{RETURN:} \quad & \texttt{emp\_last} := e.\textsf{last} \\
& \texttt{dept\_name} := d.\textsf{name} \\
& \texttt{diff} := d.\textsf{sec}.\textsf{sal} - e.\textsf{sal}
\end{aligned}$$

In this query, Q_f is the context $(e : \texttt{Emp}, d : \texttt{Dept})$, Q_w is the set containing the two equations at the WHERE clause, Q_a is the context $(\texttt{emp\_last} : \textsf{Str}, \texttt{dept\_name} : \textsf{Str}, \texttt{diff} : \textsf{Int})$, and $Q_r : Q_f \to Q_a$ is the context morphism (Definition 4.11) displayed in the RETURN clause.

9.4. QUERY BIMODULES. Any query Q on a schema **S** gives rise to a schema **R** and bimodule $\mathbf{M} : \mathbf{R} \nrightarrow \mathbf{S}$. The schema **R** is free on the schema signature (G, Υ), where G is the graph with one node $*$ and no edges, and Υ has one function symbol, i.e. generating attribute $x : * \to \tau$, for each variable $x : \tau$ in Q_a. Note that the entity category $R_\textrm{e}$ is terminal, hence $R_\textrm{o} : R_\textrm{e} \nrightarrow \mathbf{Type}$ can be identified with a single **Type**-algebra, the free algebra $R_\textrm{o} = \kappa[Q_a]$. We may refer to **R** as the *result schema*.

Using Proposition 8.24, the data of any $\mathbf{M} : \mathbf{R} \nrightarrow \mathbf{S}$ is equivalent to a single **S**-instance $M_\textrm{o}(*)$ denoted $M_\textrm{o}$, together with a morphism of **Type**-algebras $M_\textrm{r} : \kappa[Q_a] \to (M_\textrm{o})_\textrm{t}$. Equivalently, by $\Sigma_{!_S} \dashv \Delta_{!_S}$ this is a morphism of **S**-instances $M_\textrm{r} : \Sigma_{!_S} \kappa[Q_a] = \langle Q_a \rangle \to M_\textrm{o}$, using Remark 7.5 and Definition 6.13.

We thus define $M_\textrm{o} = \langle Q_f \rangle / Q_w$, precisely presented by the first two clauses of the flower syntax, while $M_\textrm{r}$ is given by the context morphism Q_r of the last clause (see Definition 4.32). Following standard database theory, we refer to $M_\textrm{o} = \langle Q_f \rangle / Q_w$ as the *frozen instance* of the query Q.

The bimodule **M** associated to Q in turn determines a functor $\Gamma_M : \mathbf{S}\text{-}\mathbf{Inst} \to \mathbf{R}\text{-}\mathbf{Inst}$; we will abuse notation by writing it as Γ_Q. It is this functor which carries out the operation of "querying an **S**-instance using Q". As the result schema **R** has a single entity, the output of this functor can be seen as a single table containing the results of the query, with one column for each variable in Q_a.

9.5. EXAMPLE. Let **S** and Q be as in Example 9.3. The query Q determines a schema **R** and a bimodule $\mathbf{M} : \mathbf{R} \nrightarrow \mathbf{S}$ as follows. The schema **R** has a single entity — call it "$*$" — and attributes $\texttt{emp\_last}, \texttt{dept\_name} : * \to \textsf{Str}$, and $\texttt{diff} : * \to \textsf{Int}$ coming from Q_a.

The bimodule **M** is determined by the frozen instance $M_\textrm{o} = \langle e : \texttt{Emp}, d : \texttt{Dept} \rangle / Q_w$, where Q_w contains the two equations from Example 9.3, together with the morphism

$$M_\textrm{r} : \langle \texttt{emp\_last}, \texttt{dept\_name} : \textsf{Str}, \texttt{diff} : \textsf{Int} \rangle \to \langle e : \texttt{Emp}, d : \texttt{Dept} \rangle / Q_w$$

given by the context morphism

$$[\texttt{emp\_last} \coloneqq e.\texttt{last}, \quad \texttt{dept\_name} \coloneqq d.\texttt{name}, \quad \texttt{diff} \coloneqq d.\texttt{sec.sal} - e.\texttt{sal}].$$

Note that the schema $\mathbf{R}$ is isomorphic to the one from Example 5.14, and that the frozen instance M_o is the instance from Example 6.16.

Let $\mathbf{J} \in \mathbf{S\text{-}Inst}$ be the instance from Example 6.3. We will now compute the result $\Gamma_Q(\mathbf{J}) \in \mathbf{R\text{-}Inst}$ of querying $\mathbf{J}$ with Q. On the single entity of $\mathbf{R}$, we have by (37) that $(\Gamma_Q J)(*) = \mathbf{S\text{-}Inst}(M_\mathrm{o}, \mathbf{J})$, and we saw in Example 6.17 that this set has three elements. The $\mathbf{Type}$-algebra $(\Gamma_Q J)_\mathrm{t}$ is the same as J_t, by (38). The values of the attributes of $\mathbf{R}$ are determined using the morphism M_r:

$$(\Gamma_Q J)(*) = \mathbf{S\text{-}Inst}\big(\langle Q_f \rangle / Q_w,\ \mathbf{J}\big) \longrightarrow \mathbf{S\text{-}Inst}\big(\langle Q_a \rangle,\ \mathbf{J}\big) \cong \prod_{(x:\tau) \in Q_a} J_\mathrm{t}(\tau). \qquad (49)$$

A transform $\langle Q_f \rangle \to \mathbf{J}$ has an underlying context morphism $\Phi \to Q_f$, where Φ is the context of the canonical presentation of $\mathbf{J}$ (see Remark 6.15). We can express (49) using context morphisms: given a transform $\langle Q_f \rangle / Q_w \to \mathbf{J}$ corresponding to an element of $(\Gamma_Q J)(*)$, simply compose its underlying context morphism $\Phi \to Q_f$ with $Q_r \colon Q_f \to Q_a$. The attributes of this row of the table "$*$" can be read off of the resulting context morphism $\Phi \to Q_a$.

Doing this, we obtain the result

$*$	emp_last	dept_name	diff
1	Noether	HR	100
2	Euclid	HR	150
3	Euclid	Admin	0

(Note that the row-ids are arbitrary.) For example, the first row corresponds to the transform $\langle Q_f \rangle \to \mathbf{J}$ given by $[e \coloneqq e2, d \coloneqq d1]$. Composing this with Q_r gives $[\texttt{emp\_last} \coloneqq e2.\texttt{last}, \texttt{dept\_name} \coloneqq d1.\texttt{name}, \texttt{diff} \coloneqq d1.\texttt{sec.sal} - e1.\texttt{sal}]$, which simplifies to the first row of the table above.

9.6. REMARK. By Corollary 8.34, the result of any query Q on $\mathbf{S}$, with result schema $\mathbf{R}$ and associated bimodule $\mathbf{M}$, is equivalently obtained as the composite of data migration functors

$$\mathbf{S\text{-}Inst} \xrightarrow{\ \Pi_{i_\mathbf{S}}\ } \mathbf{Col(M)\text{-}Inst} \xrightarrow{\quad \Delta_{i_\mathbf{R}}\quad} \mathbf{R\text{-}Inst}$$

$$\mathbf{J} \longmapsto \Pi_{i_\mathbf{S}}(\mathbf{J}) \longmapsto \Delta_{i_\mathbf{R}}\big(\Pi_{i_\mathbf{S}}(\mathbf{J})\big) \cong \Gamma_Q(\mathbf{J}) \qquad (50)$$

where the schema mappings $\mathbf{i_R} \colon \mathbf{R} \to \mathbf{Col(M)} \leftarrow \mathbf{S} \colon \mathbf{i_S}$ into the bimodule collage $\mathbf{Col(M)}$ are as in Section 8.29.

For example, the query in Example 9.5 gave the same result as we found using Examples 7.8 and 7.10. One can check that the bimodule collage is $\mathbf{Col(M)} \cong \mathbf{T}$ given in (25), and the mappings $\mathbf{F}$, $\mathbf{G}$ of the mentioned examples are the collage inclusions. Hence this is an instance of (50).

9.7. REMARK. In practice, one would like a guarantee that a query result $\Gamma_Q(\mathbf{J})$ is finite whenever $\mathbf{J}$ is finite. To achieve this, one has to place the extra condition on the query Q that only entities — no types — appear in Q_f. This condition also ensures the *domain independence* [AHV95] of the query, meaning that it is not necessary to enumerate the elements of a type to compute the query result.

9.8. UBER-QUERIES. If queries correspond to $(\mathbf{R}, \mathbf{S})$-bimodules where $\mathbf{R}$ has only one entity, then we need a name for more general bimodules; we call them *uber-queries*. An uber-query is roughly a diagram of queries. The morphisms in this diagram will be called Keys, and our syntax is accordingly extended to be of the form For-Where-Keys-Return.

9.9. EXAMPLE. To describe a bimodule of the following form
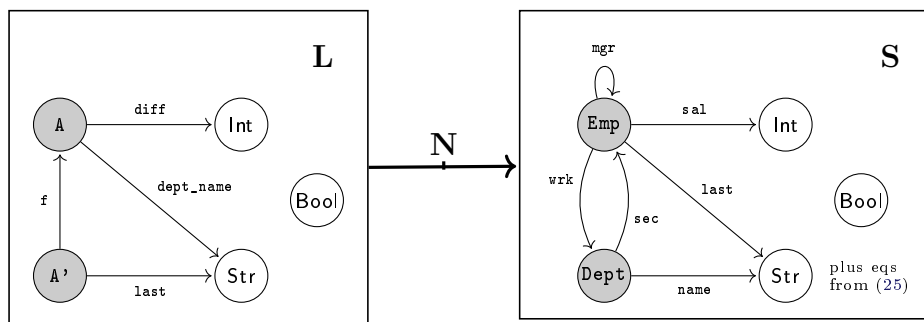
we will need two instances $\mathbf{I} := N(\mathtt{A})$ and $\mathbf{I}' := N(\mathtt{A}')$, and a transform $N(f)\colon \mathbf{I}' \to \mathbf{I}$ between them, as well as three terms $\mathtt{diff}, \mathtt{name}, \mathtt{last}$ of the specified types in $\mathbf{I}$ and $\mathbf{I}'$. Indeed, this gives a functor $\widetilde{L}^{\mathrm{op}} \to \mathbf{S}\text{-}\mathbf{Inst}$ (where objects Int, Str, and Bool in the type side are sent to the corresponding representable instances, as usual; Definition 8.3).

In Example 6.18, we constructed two $\mathbf{S}$-instances and a transform $\mathbf{I}' \to \mathbf{I}$ between them . We will rewrite them, together with the three terms, in For-Where-Keys-Return syntax below.

$$\mathtt{A}' =$$

FOR: $e' : \mathtt{Emp}$
WHERE: $e'.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin}$
$\quad e'.\mathtt{sal} \leq e'.\mathtt{wrk}.\mathtt{sec}.\mathtt{sal}$
KEYS: $\mathtt{f} := \mathtt{Emp}[e := e', d := e'.\mathtt{wrk}]$
RETURN: $\mathtt{last} := e'.\mathtt{last}$

$$\mathtt{A} =$$

FOR: $e : \mathtt{Emp}, d : \mathtt{Dept}$
WHERE: $e.\mathtt{wrk}.\mathtt{name} = \mathrm{Admin}$
$\quad e.\mathtt{sal} \leq d.\mathtt{sec}.\mathtt{sal}$
RETURN: $\mathtt{dept\_name} := e.\mathtt{wrk}.\mathtt{name}$
$\quad \mathtt{diff} := d.\mathtt{sec}.\mathtt{sal} - e.\mathtt{sal}$

For any $\mathbf{S}$-instance $\mathbf{J}$, we can apply $\Gamma_N \colon \mathbf{S}\text{-}\mathbf{Inst} \to \mathbf{L}\text{-}\mathbf{Inst}$. If $\mathbf{J}$ is as in Example 6.3, then $\Gamma_N(\mathbf{J})$ is the following $\mathbf{L}$-instance:

A'	last	f
0	Euclid	3

A	dept_name	diff
1	HR	100
2	HR	150
3	Admin	0

10. Implementation

The mathematics developed in this paper has been implemented using OPL, an Operadic Programming Language, which can be can be downloaded from http://categoricaldata. net/fql.html. The examples in this paper are pre-loaded as the "Paper" example in the software package. In this section we briefly discuss implementation issues that arise, namely in negotiating between syntactic presentations (e.g. those discussed in Section 4) and the objects they denote.

Most constructions involving finitely-presented categories, including query evaluation and collage construction, depend crucially on solving word problems in categories, and these problems are not in general decidable. In Section 10.1 we describe our approach to solving word problems, and in Section 10.2 we describe how we use word problems to compute collages and evaluate queries.

10.1. SOLVING WORD PROBLEMS. Given a category presentation (G, E) as described in Section 4.35, the word problem is to decide if two terms (words) in G are equivalent under E. The word problem is obviously semi-decidable: to prove if two terms p and q in G are equal under E, we can systematically enumerate all of the (usually infinite) consequences of E until we find $p = q$. However, if p and q are not equal, then this enumeration will never stop. In practice, not only is enumeration computationally infeasible, but for query evaluation and collage construction, we require a true decision procedure: an algorithm which, when given p and q as input, will always terminate with "equal" or "not equal". Hence, we must look to efficient, but incomplete, automated theorem proving techniques to decide word problems.

The OPL tool allows any theorem prover to be used to decide word problems. In addition, the OPL tool also provides a default, built-in theorem prover based on Knuth-Bendix completion [KB70]: from (Σ, E), it attempts to construct a system of re-write rules (oriented equations), R, such that p and q are equal under E if and only if p and q re-write to syntactically equal terms (so-called *normal forms*) under R. We demonstrate this with an example. Consider a presentation of the algebraic theory of groups, on the left, below. Knuth-Bendix completion yields the re-write system on the right, below: [7]

---

[7]Because there is only one sort, say $S_\Sigma = \{G\}$, we drop the contexts in the Axiom side. For example, the second equation — *axiom* — should be $x : G \vdash (x^{-1} * x = 1) : G$, according to Section 4.

$$\begin{array}{ll}
\text{Axioms} & \text{Re-write rules} \\
1 * x = x & 1 * x \rightsquigarrow x \\
x^{-1} * x = 1 & x^{-1} * x \rightsquigarrow 1 \\
(x * y) * z = x * (y * z) & (x * y) * z \rightsquigarrow x * (y * z) \\
 & x^{-1} * (x * y) \rightsquigarrow y \\
 & 1^{-1} \rightsquigarrow 1 \\
 & x * 1 \rightsquigarrow x \\
 & (x^{-1})^{-1} \rightsquigarrow x \\
 & x * x^{-1} \rightsquigarrow 1 \\
 & x * (x^{-1} * y) \rightsquigarrow y \\
 & (x * y)^{-1} \rightsquigarrow y^{-1} * x^{-1}
\end{array}$$

To see how these re-write rules are used to decide the word problem, consider the two terms $(a^{-1} * a) * (b * b^{-1})$ and $b * ((a * b)^{-1} * a)$. Both of these terms re-write to 1 under the above re-write rules; hence, we conclude that they are equal. In contrast, the two terms $1 * (a * b)$ and $b * (1 * a)$ re-write to $a * b$ and $b * a$, respectively, which are not syntactically the same; hence, we conclude that they are not equal.

The details of how the Knuth-Bendix algorithm works are beyond the scope of this paper. However, we make two remarks. First, Knuth and Bendix's original algorithm ([KB70]) can fail even when a re-write system to decide a word problem exists; for this reason, we use the more modern, "unfailing" variant of Knuth-Bendix completion [BDP89]. Second, we remark that Buchberger's algorithm for computing Gröbner bases is a very similar algorithm used in many computer algebra systems, and it may be seen as the instantiation of the Knuth-Bendix algorithm in the theory of polynomial rings [Mar96].

10.2. SATURATION AND QUERY EVALUATION. Given a category presentation (G, E), a decision procedure for the word problem allows us to (semi) compute the category $\mathscr{C}$ that (G, E) presents. To do this, we construct $\mathscr{C}$ in stages: first, we find all non-equal terms of size 0 in G; [8] call this $\mathscr{C}^0$. Then, we add to $\mathscr{C}^0$ all non-equal terms of size 1 that are not equal to a term in $\mathscr{C}^0$; call this $\mathscr{C}^1$. We iterate this procedure, obtaining a sequence $\mathscr{C}^0, \mathscr{C}^1, \ldots$. If $\mathscr{C}$ is indeed finite, then there will exist some n such that $\mathscr{C}^n = \mathscr{C}^{n+1} = \mathscr{C}$ and we can stop. Otherwise, our attempt to construct $\mathscr{C}$ will run forever: it is not decidable whether a given presentation (G, E) generates a finite category. The category $\mathscr{C}$ will be isomorphic to the category $\mathrm{Fr}(G)/E$ obtained by quotienting the free category on G by the equations in E (Definition 4.41); essentially, $\mathscr{C}$ represents equivalence classes of terms by a smallest possible representative, as explained in detail in Section 4. (Note that the normal forms chosen by the internal Knuth-Bendix theorem prover for the purposes of deciding the word problem need not in general be the same as the chosen representatives of equivalences classes in $\mathscr{C}$.)

Most uses of the OPL tool involve *saturating* instance presentations into collages so that they may be examined as tables (e.g. Example 6.16). This just means replacing part

---

[8]By the *size* of a term, we mean the height of the associated syntax tree. For example $\max(x.\mathtt{sal}, x.\mathtt{mgr.sal})$ has size of three.
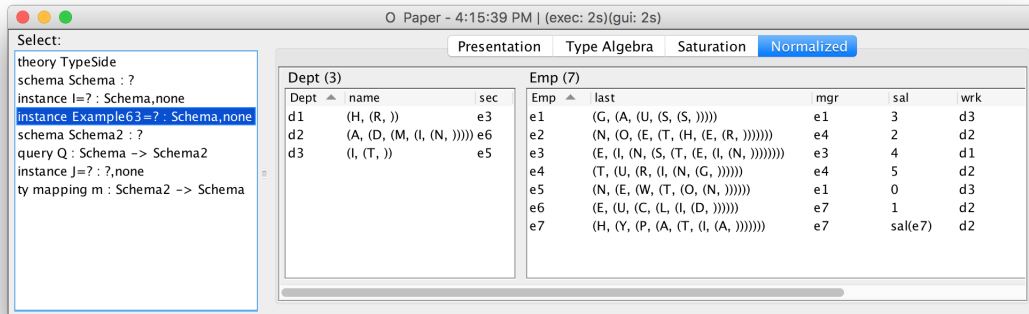
Figure 7: OPL displaying the instance from Example 6.3

of the presentation with a canonical presentation (see Remarks 4.21, 4.34 and 6.15[9]) The saturation process is very similar to the process described in the preceding paragraph, with one small difference. In general, the 'type side' of the collage (see Remark 5.3) will denote an infinite category. For example, if **Type** is the free group on one generator $\{a\}$, it will contain $a, a*a, a*a*a$, and so on. Hence, it is usually not possible to saturate the type side of an instance. So, the OPL tool saturates only the entity side of an instance, which will often be finite in practice. From the saturated entity side presentation and a set of re-write rules for the collage, it is possible to construct a set of tables that faithfully represent the instance. The tables for the entity side of the instance are simply a tabular rendering of the finite category corresponding to the saturated entity side of the instance's collage ($\mathscr{C}$ in the preceding paragraph). The tables for the attributes of the instance must also contain representatives of equivalence classes of terms, but unlike the entity side case, where representatives are chosen based on size, it is less clear which representative to choose. For example, there is an implicit preference to display 1,024 instead of 2^{10}, even though the size (as defined in the previous footnote) of the former is greater than the size of the latter. The OPL tool allows these representatives to be computed by external programs, thereby providing a "hook" for the tool to interface with other programming languages and systems. For example, users can provide a Java implementation of natural numbers for the commutative ring type side used in this paper, and the java compiler will normalize terms like 2^{10} into 1024. By default, the OPL tool will display the normal forms computed by the internal Knuth-Bendix theorem prover in the attribute tables.

To evaluate a query Q such as that in Example 9.5 on a presented instance $\mathbf{I}$, we first saturate the entity side of $\mathbf{I}$ as described in the preceding paragraph. Evaluation of the query, $\Gamma_Q(\mathbf{I})$ as in Section 9.4, proceeds similarly to evaluation of 'For-Where-Return' queries in traditional SQL systems [AHV95]: first, we compute a (typically large) set of tuples corresponding to the FOR clause by repeatedly looping through $\mathbf{I}$. Then, we filter

---

[9]For explanatory reasons these particular examples saturate a frozen instance associated with a query, but the implementation does not need to saturate frozen instances to evaluate queries.

this set of tuples by the WHERE clause; here we must be sure to decide equality of tuples under the equational theory for **I**, using Knuth-Bendix as described above. Finally, we project out certain components of these tuples, according to the RETURN clause. The result of the query will be a saturated instance, which has a canonical presentation as in Remark 6.15.

The OPL tool's tabular rendering of the instance from Example 6.3 is shown in Fig. 7. Because a unary representation of the integers is computationally inefficient, for expediency the employee salaries in the OPL program have been reduced compared to Example 6.3. A more efficient axiomatization of the integers, such as using binary, can also be used.

A. Componentwise composition and exponentiation in $\mathbb{D}\mathbf{ata}$

We defined composition of bimodules and 2-cells in (40) and (41) and exponentiation of bimodules in Proposition 8.16. In Proposition 8.20 we saw that bimodules can be equivalently defined in several components, separating the entity and type sides of the structure. It is natural to ask what composition and exponentiation (and as special cases, the data migration functors) look like in this decomposed view.

In fact, when first working out the ideas presented in this paper, we used componentwise formulas to understand all the constructions. In writing it up, we decided that the coend formulas were more succinct and often easier to work with; however, the machinery below still turns out to be useful in certain cases, so we present it without proof for the interested reader.

Recall the left tensor $\otimes$ defined in Definition 3.11, which 'preserves algebraicity' of profunctors on the right.

A.1. PROPOSITION. *The composition* $\mathbf{M} \odot \mathbf{N}$ *of two bimodules* $\mathbf{R} \xrightarrow{\mathbf{M}} \mathbf{S} \xrightarrow{\mathbf{N}} \mathbf{T}$ *in* $\mathbb{D}\mathbf{ata}$ (40) *is equivalently given in components as follows:* $(M \odot N)_\mathrm{e} = M_\mathrm{e} \odot N_\mathrm{e}$ *in* $\mathbf{Prof}$, *and the rest of the components are given using a pushout, as in the following diagram in the category* $[R_\mathrm{e}^\mathrm{op}, \mathbf{Type\text{-}Alg}]$:
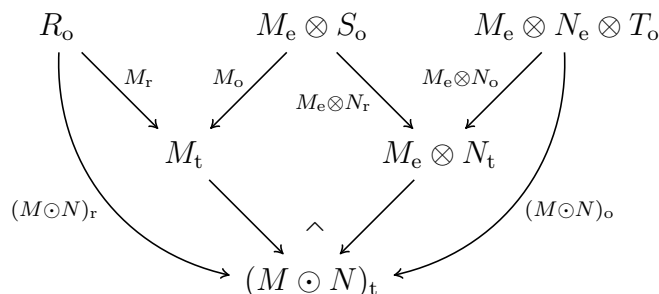
This follows from the following lemma, which can be proven using Proposition 2.25:

A.2. LEMMA. *Let $L\colon A_0 \twoheadrightarrow A_1$, $M\colon B_0 \twoheadrightarrow B_1$, and $N\colon C_0 \twoheadrightarrow C_1$ be proarrows in an equipment $\mathbb{D}$ with extensive collages and local finite colimits. Let $X \in {}_L\mathbf{Simp}_M$ and $Y \in {}_M\mathbf{Simp}_N$ be simplices, and let $P\colon \widetilde{L} \twoheadrightarrow \widetilde{M}$ and $Q\colon \widetilde{M} \twoheadrightarrow \widetilde{N}$ be proarrows such that $X \cong {}_L\mathrm{Res}_M(P)$ and $Y \cong {}_M\mathrm{Res}_N(Q)$ (see Section 2.22). Then the components of ${}_L\mathrm{Res}_N(P \odot Q)$ can be computed by pushout:*

$$
\begin{array}{ccc}
X_{i,0} \odot M \odot Y_{1,j} & \longrightarrow & X_{i,0} \odot Y_{0,j} \\
\downarrow & & \downarrow \\
X_{i,1} \odot Y_{1,j} & \longrightarrow & \widehat{i}_{A_i} \odot (P \odot Q) \odot \widecheck{i}_{C_j}
\end{array}
\tag{51}
$$

Moreover, the 2-cells of ${}_L\mathrm{Res}_N(P \odot Q)$ are found using these pushouts in an evident way.

A.3. PROPOSITION. *The horizontal composition of 2-cells in $\mathbb{D}\mathbf{ata}$*

$$
\begin{array}{ccccc}
\mathbf{R} & \xrightarrow{\ \mathbf{M}\ } & \mathbf{S} & \xrightarrow{\ \mathbf{N}\ } & \mathbf{T} \\
{\scriptstyle\mathbf{F}}\downarrow & \Downarrow\boldsymbol{\theta} & \downarrow{\scriptstyle\mathbf{G}} \ \ \Downarrow\boldsymbol{\phi} & & \downarrow{\scriptstyle\mathbf{H}} \\
\mathbf{R}' & \xrightarrow[\ \mathbf{M}'\ ]{} & \mathbf{S}' & \xrightarrow[\ \mathbf{N}'\ ]{} & \mathbf{T}'
\end{array}
$$

is given (Proposition 8.22) by $(\theta \odot \phi)_{\mathrm{e}} = \theta_{\mathrm{e}} \odot \phi_{\mathrm{e}}$, while $(\theta \odot \phi)_{\mathrm{t}}$ is induced by the diagram

$$
\begin{array}{ccccc}
M_{\mathrm{t}} & \xleftarrow{\ M_{\mathrm{o}}\ } & M_{\mathrm{e}} \otimes S_{\mathrm{o}} & \xrightarrow{\ M_{\mathrm{e}} \otimes N_{\mathrm{r}}\ } & M_{\mathrm{e}} \otimes N_{\mathrm{t}} \\
{\scriptstyle\theta_{\mathrm{t}}}\downarrow & & \downarrow{\scriptstyle\theta_{\mathrm{e}} \otimes G_{\mathrm{o}}} & & \downarrow{\scriptstyle\theta_{\mathrm{e}} \otimes \phi_{\mathrm{t}}} \\
\widehat{F}_{\mathrm{e}} \otimes M'_{\mathrm{t}} & \xleftarrow[\ \mathrm{id} \otimes M'_{\mathrm{o}}\ ]{} & \widehat{F}_{\mathrm{e}} \otimes M'_{\mathrm{e}} \otimes S'_{\mathrm{o}} & \xrightarrow[\ \mathrm{id} \otimes M'_{\mathrm{e}} \otimes N'_{\mathrm{r}}\ ]{} & \widehat{F}_{\mathrm{e}} \otimes M'_{\mathrm{e}} \otimes N'_{\mathrm{t}}
\end{array}
$$

where $(M \odot N)_{\mathrm{t}}$ and $(M' \odot N')_{\mathrm{t}}$ are the pushouts of the top and bottom rows respectively, by Proposition A.1.

A.4. PROPOSITION. *Let $\mathbf{N}\colon \mathbf{S} \twoheadrightarrow \mathbf{T}$ and $\mathbf{P}\colon \mathbf{R} \twoheadrightarrow \mathbf{T}$ be bimodules. The bimodule $\mathbf{N} \triangleright \mathbf{P}\colon \mathbf{R} \twoheadrightarrow \mathbf{S}$ is given as follows: the entity component $(N \triangleright P)_{\mathrm{e}}$ is computed by a pointwise pullback, for any objects $s \in S_{\mathrm{e}}$, $r \in R_{\mathrm{e}}$,*

$$
\begin{array}{ccc}
(N \triangleright P)_{\mathrm{e}}(r, s) & \longrightarrow & \mathbf{Set}^{T_{\mathrm{e}}}\big(N_{\mathrm{e}}(s, -), P_{\mathrm{e}}(r, -)\big) \\
\downarrow & \lrcorner & \downarrow{\scriptstyle P_{\mathrm{o}}} \\
& & \mathbf{Set}^{T_{\mathrm{e}}}\Big(N_{\mathrm{e}}(s, -), \mathbf{Type\text{-}Alg}\big(T_{\mathrm{o}}(-), P_{\mathrm{t}}(r)\big)\Big) \\
\downarrow & & \downarrow{\scriptstyle\cong} \\
\mathbf{Type\text{-}Alg}\big(N_{\mathrm{t}}(s), P_{\mathrm{t}}(r)\big) & \xrightarrow{\ N_{\mathrm{o}}\ } & \mathbf{Type\text{-}Alg}\big((N_{\mathrm{e}} \otimes T_{\mathrm{o}})(s), P_{\mathrm{t}}(r)\big).
\end{array}
$$

Equivalently, $(N \triangleright P)_{\mathrm{e}}(r, s) = \mathbf{T\text{-}Inst}\big(N(s), P(r)\big)$. The other components are $(N \triangleright P)_{\mathrm{t}} = P_{\mathrm{t}}$, $(N \triangleright P)_{\mathrm{r}} = P_{\mathrm{r}}$, and $(N \triangleright P)_{\mathrm{o}}$ is the composition

$$
(N \triangleright P)_{\mathrm{e}} \otimes S_{\mathrm{o}} \to \mathbf{Type\text{-}Alg}\big(N_{\mathrm{t}}(-), P_{\mathrm{t}}(-)\big) \otimes N_{\mathrm{t}} \to P_{\mathrm{t}}.
$$

References

[AHV95]   Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[AR94]   Jiří Adámek and Jiří Rosický. *Locally presentable and accessible categories*. London Mathematical Society Lecture Note Series 189. Cambridge University Press, 1994. ISBN: 0-521-42261-2.

[ARV11]   J. Adámek, J. Rosický, and E. M. Vitale. *Algebraic theories*. Cambridge Tracts in Mathematics 184. Cambridge University Press, 2011. ISBN: 978-0-521-11922-1.

[BDP89]   L. Bachmair, N. Dershowitz, and D.A. Plaisted. "Completion Without Failure". In: *Resolution of Equations in Algebraic Structures - Rewriting Techniques* 2 (1989), pp. 1–30.

[Bén00]   Jean Bénabou. "Distributors at work". Lecture notes written by Thomas Streicher. 2000. URL: http://www.mathematik.tu-darmstadt.de/~streicher/FIBR/DiWo.pdf.

[Bor94a] Francis Borceux. *Handbook of categorical algebra*. Vol. 1: *Basic category theory*. 3 vols. Encyclopedia of Mathematics and its Applications 50. Cambridge University Press, 1994. ISBN: 0-521-44178-1.

[Bor94b] Francis Borceux. *Handbook of categorical algebra*. Vol. 2: *Categories and structures*. 3 vols. Encyclopedia of Mathematics and its Applications 51. Cambridge University Press, 1994. ISBN: 0-521-44178-1.

[BW85]   Michael Barr and Charles Wells. *Toposes, Triples and Theories*. A Series of Comprehensive Studies in Mathematics 278. Springer-Verlag, 1985. ISBN: 0-387-96115-1.

[DHI12]   AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN: 0124160441, 9780124160446.

[FGR03]   Michael Fleming, Ryan Gunther, and Robert Rosebrugh. "A database of categories". In: *Journal of Symbolic Computation* 35.2 (Feb. 2003), pp. 127–135. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(02)00104-9.

[GP04]   Marco Grandis and Robert Paré. "Adjoint for double categories". In: *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 45.3 (2004), pp. 193–240.

[GS15]   Richard Garner and Michael Shulman. *Enriched categories as a free cocompletion*. Nov. 2015. arXiv: 1301.3191v2 [math.CT].

[Hun04]   Edward V. Huntington. "Sets of independent postulates for the algebra of logic". In: *Transactions of the American Mathematical Society* 5.3 (1904), pp. 288–309. DOI: 10.1090/s0002-9947-1904-1500675-4.

618　　　　　SCHULTZ, SPIVAK, VASILAKOPOULOU, WISNESKY

[Jac99]　　Bart Jacobs. *Categorical logic and type theory*. Vol. 141. Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Co., Amsterdam, 1999, pp. xviii+760. ISBN: 0-444-50170-3.

[JR02]　　Michael Johnson and Robert Rosebrugh. "Sketch Data Models, Relational Schema and Data Specifications". In: *Electronic Notes in Theoretical Computer Science* 61 (2002). CATS'02, Computing: the Australasian Theory Symposium, pp. 51–63. ISSN: 1571-0661.

[KB70]　　Donald Knuth and Peter Bendix. "Simple Word Problems in Universal Algebra". In: *Computational Problems in Abstract Algebra*. Ed. by John Leech. Pergamon Press, 1970, pp. 263–297.

[Kel05]　　G. M. Kelly. "Basic concepts of enriched category theory". In: *Reprints in Theory and Applications of Categories* 10 (2005). URL: http://www.tac.mta.ca/tac/reprints/articles/10/tr10abs.html.

[Kel89]　　G.M. Kelly. "Elementary observations on 2-categorical limits". In: *Bulletin of the Australian Mathematical Society* 39.02 (Apr. 1989), pp. 301–317. ISSN: 1755-1633. DOI: 10.1017/S0004972700002781.

[Lei04]　　Tom Leinster. *Higher operads, higher categories*. London Mathematical Society Lecture Note Series 298. Cambridge University Press, Cambridge, 2004. ISBN: 0-521-53215-9. DOI: 10.1017/CBO9780511525896.

[Mac98]　　Saunders Mac Lane. *Categories for the working mathematician*. 2nd ed. Graduate Texts in Mathematics 5. New York: Springer-Verlag, 1998. ISBN: 0-387-98403-8.

[Mar96]　　Claude Marché. "Normalized Rewriting: An Alternative to Rewriting Modulo a Set of Equations". In: *Journal of Symbolic Computation* 21.3 (Mar. 1996), pp. 253–288. ISSN: 0747-7171. DOI: 10.1006/jsco.1996.0011.

[Mat89]　　Hideyuki Matsumura. *Commutative ring theory*. Trans. from Japanese by M. Reid. 2nd ed. Cambridge Studies in Advanced Mathematics 8. Cambridge University Press, 1989. ISBN: 0-521-36764-6.

[Mit96]　　John C. Mitchell. *Foundations of Programming Languages*. Cambridge, MA: MIT Press, 1996. ISBN: 0-262-13321-0.

[MS08]　　Katherine Munn and Barry Smith. *Applied ontology: an introduction*. Vol. 9. Walter de Gruyter, 2008.

[nLab]　　*Nerve and realization*. URL: http://ncatlab.org/nlab/show/nerve+and+realization (visited on 01/19/2016).

[RW92]　　Robert Rosebrugh and R. J. Wood. "Relational Databases and Indexed Categories". In: *Canadian Mathematical Society Conference Procedings*. International Summer Category Theory Meeting. (June 23–30, 1991). Ed. by R. A. G. Seely. Vol. 13. American Mathematical Society, 1992, pp. 391–407.

[Sch15]   Patrick Schultz. *Regular and exact (virtual) double categories*. 2015. arXiv: 1505.00712 [math.CT].

[Shu08]   Michael Shulman. "Framed bicategories and monoidal fibrations". In: *Theory and Applications of Categories* 20.18 (2008), pp. 650–738. arXiv: 0706.1286 [math.CT].

[Shu10]   Michael Shulman. *Constructing symmetric monoidal bicategories*. 2010. arXiv: 1004.0993 [math.CT].

[Spi12]   David I. Spivak. "Functorial data migration". In: *Information and Computation* 127 (Aug. 2012), pp. 31–51. ISSN: 0890-5401. DOI: 10.1016/j.ic.2012.05.001. arXiv: 1009.1166v4 [cs.DB].

[SW15]   David I. Spivak and Ryan Wisnesky. "Relational Foundations for Functorial Data Migration". In: *Proceedings of the 15th Symposium on Database Programming Languages*. DBPL 2015. Pittsburgh, PA: ACM, 2015, pp. 21–28. ISBN: 978-1-4503-3902-5. DOI: 10.1145/2815072.2815075.

[Web07]   Mark Weber. "Familial 2-Functors and Parametric Right Adjoints". In: *Theory and Applications of Categories* 18.22 (2007), pp. 665–732.

[Wis+15]   Ryan Wisnesky, David I. Spivak, Patrick Schultz, and Eswaran Subrahmanian. *Functorial Data Migration: From Theory to Practice*. report G2015-1701. National Institute of Standards and Technology, 2015. arXiv: 1502.05947v2.

[Woo82]   R. J. Wood. "Abstract pro arrows I". In: *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 23.3 (1982), pp. 279–290. ISSN: 0008-0004.

[Woo85]   R. J. Wood. "Proarrows II". In: *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 26.2 (1985), pp. 135–168.

Massachusetts Institute of Technology, 77 Massachusetts Ave. Cambridge, MA 02139
Email: schultzp@mit.edu
        dspivak@mit.edu,
        cvasilak@mit.edu
        ryan@catinf.com