logics

Article

Graph Algebras and Derived Graph Operations

Uwe Wolter * and Tam T. Truong

Department of Informatics, University of Bergen, 5020 Bergen, Norway; tam.truong@uib.no
* Correspondence: uwe.wolter@uib.no

Abstract: We revise our former definition of graph operations and correspondingly adapt the construction of *graph term algebras*. As a first contribution to a prospective research field, *Universal Graph Algebra*, we generalize some basic concepts and results from algebras to graph algebras. To tackle this generalization task, we revise and reformulate traditional set-theoretic definitions, constructions and proofs in Universal Algebra by means of more category-theoretic concepts and constructions. In particular, we generalize the concept of *generated subalgebra* and prove that all monomorphic homomorphisms between graph algebras are regular. *Derived graph operations* are the other main topic. After an in-depth analysis of *terms* as representations of derived operations in traditional algebras, we identify three basic mechanisms to construct new graph operations out of given ones: parallel composition, instantiation, and sequential composition. As a counterpart of terms, we introduce *graph operation expressions* with a structure as close as possible to the structure of terms. We show that the three mechanisms allow us to construct, for any graph operation expression, a corresponding *derived graph operation* in any graph algebra.

Keywords: graph operation; graph algebra; graph term algebra; Lawvere theory; derived graph operation; graph operation expression; universal graph algebra; string diagrams

Citation: Wolter, U.; Truong, T.T. Graph Algebras and Derived Graph Operations. *Logics* **2023**, *1*, 182–239. https://doi.org/10.3390/logics1040010

Academic Editors: Cristina Sernadas and João Rasga

Received: 12 May 2023
Revised: 28 August 2023
Accepted: 11 September 2023
Published: 17 October 2023

1. Introduction

The paper is a relatively independent part of a broader long-term project to develop a proper foundation of diagrammatic specification formalisms and diagrammatic logics. The project is centered around and extends the concept of *generalized sketches*. We use the term "diagrammatic" as a synonym for "graph-based" in a very broad sense including arbitrary presheaf topoi.

One of the objectives of our project is to elevate traditional first-order logic to a wide range of arbitrary categories. In [1], we only addressed predicates and showed how to define corresponding first-order *logics of statements in context* without operations in arbitrary categories. The present paper is also meant to be a first step towards an abstract notion of an operation allowing us to define fully fledged first-order *logics of statements in context*, at least, in arbitrary topoi.

Generalized sketches have been developed in the 1990s independently by Michael Makkai, motivated by his work on an abstract formulation of completeness theorems in logic [2], and a group in Latvia around Zinovy Diskin, motivated by their work on data bases and data modeling [3–5]. Our concept of *graph operation* has its origins in the concept of *sketch operation*. Sketch operations do not appear in the work of Makkai, but they have been an integral part of Diskin's pioneering work from the very beginning.

Graph operations (and their prospective generalizations) are vital in software engineering. Moreover, they provide a conceptual tool with many potential and useful applications in mathematics, logic, and computer science.

Software models are often diagrammatic structures. To keep software models comprehensible for humans, we should, however, avoid overloading them with auxiliary items and redundant information. Nonetheless, to formulate and integrate relevant constraints in software models, it is necessary to refer to items that are not present in a model but that can be derived, like the composition of references, for example. Graph operations are an appropriate tool to refer to and reason about those derivable items in diagrammatic artifacts.

Graph operations are also vital for the definition of query languages for diagrammatic data models, for example. The crucial idea is to formalize queries as *derived graph operations* built up from basic operations like the operations of Codd's relational algebra, for example. The potential to formalize query languages for diagrammatic data models was one of Diskin's main motivations to introduce sketch operations [4].

An example par excellence for the conceptual potential of graph operations in mathematics are categories. Categories can be described as graphs plus an identity and a composition operation. It is not common yet, but quite natural, to consider these operations as graph operations. The identity operation introduces, for each vertex in a graph, a loop, while the composition operation generates, for any two successive edges in a graph, a corresponding composite edge. Even statements like "we assume a category with chosen pullbacks", for example, can be adequately made precise by means of corresponding graph operations. Ref. [6] seems to be the first paper outlining the potential of graph operations in category theory. Since categories are nowadays widely used in computer science, logic, mathematics, and physics, we will use them as our running example.

In traditional string-based formalisms and logics, *terms* are the standard tool used to represent and reason about "derivable data". At the same time, terms give us an adequate tool at hand to represent *derived operations*, i.e., operations that can be built up from the basic operations in an algebra. Therefore, we generalized in [7] the construction of terms from traditional algebras to graph algebras. The construction of graph term algebras and their characterization as a free construction is the main result in [7]. Unfortunately, our expectation that *graph terms* would give us a universal and appropriate concept of *derived graph operations* and *substitutions* at hand was naive. We realized that the strong interconnection between the "representation of data" and the "representation of derived operations" breaks down in the case of graph operations.

After Section 2, where we list some basic notations, concepts, conventions, and results, the paper presents a further development of the theory of graph operations and graph algebras in two directions—model theory (including term algebras) and derived graph operations.

Model theory: In [7], we coined the concept of graph algebra, introduced *graph terms*, and showed that *graph term algebras* are free graph algebras. There was, however, no model theory in the sense of traditional Universal Algebra. As a kind of "proof of concept", we therefore generalize some basic model-theoretic concepts and results from traditional algebras to graph algebras. We concentrate on the concept of "generated subalgebra" and the related problem of characterizing monomorphic and epimorphic homomorphisms.

To tackle this task, we make a substantial effort in Section 3 to revise and reformulate traditional set-theoretic definitions, constructions and proofs in Universal Algebra by means of more category-theoretic concepts and constructions. Relying on this reformulation, we can in Section 4 smoothly transfer concepts, definitions and results from traditional algebras to graph algebras. In particular, we prove that all monomorphic homomorphisms between graph algebras are regular.

In [7], we adapted the original idea of sketch operations [4] and defined the arity of a graph operation as a single graph inclusion. This definition does not allow us, however, to consider projections as legal graph operations. To be closer to the traditional concept of *operation* in Universal Algebra, and to be able to define an appropriate concept of a *derived graph operation*, we therefore declare in this paper the arity of a graph operation as a span of graph inclusions. In Section 4.2, we clarify the relation between both versions and discuss to what extent they are equivalent.

To prove that all monomorphic homomorphisms between graph algebras are regular, we also introduce partial graph algebras in Section 4.4. We define a so-called *term completion* procedure transforming partial graph algebras into total graph algebras. This procedure provides for each signature a free functor from the corresponding category of partial graph algebras to the corresponding category of graph algebras. The construction of graph term algebras turns out to be just a special case of this new procedure.

Derived graph operations: To understand why the strong interconnection between "representation of data" and "representation of derived operations" breaks down in the case of graph operations and to find out how to define *derived graph operations* in an appropriate way, we include in the paper a more in-depth analysis of the concept *term* and discuss *substitution calculi* in general in Section 3.4.

In Section 3.7, we recall the construction of *syntactic Lawvere theories* as it is described in [8], for example. In Section 5.1, we discuss finite product categories and elucidate that terms can be characterized as *normal forms* for *finite product expressions*; thus, Lawvere's original slogan "composition is substitution" can be turned into the slogan "substitution is symbolic composition plus normalization".

Reviewing the relationship between *finite products* and *tensor products*, we identify *copying* in Section 5.2 as the cause of the problem. We argue that, in the case of graph operations, "copying of data", as a computation of its own, has to be replaced by the "soldering" of input and (!) output ports of computations.

In such a way, we end up in Section 5.3 with three mechanisms to construct new graph operations out of given ones: parallel composition, instantiation ("soldering" of input and output ports), and sequential composition.

Finally, we introduce in Section 5.4 *graph operation expressions* with a structure as close as possible to the structure of terms. We define their semantics, i.e., the *derived graph operations* we have been looking for, by means of the three mechanisms of parallel composition, instantiation and sequential composition.

We complete the paper with some remarks concerning *Operations in Topoi* in Section 6, a discussion of *Related Work* in Section 7 and concluding remarks in Section 8.

2. Notations and Preliminaries

C_{Obj} denotes the collection of objects of a category C and C_{Mor} as the collection of morphisms of C, respectively. $\mathsf{C}(A, B)$ is the collection of all morphisms from object A to object B in C. If the category C is clear from the context, we will often use the more compact notation B^A instead of $\mathsf{C}(A, B)$. We use the diagrammatic notation $f;g : A \to C$ for the composition of morphisms $f : A \to B$ and $g : B \to C$ in C. $\mathsf{C} \sqsubseteq \mathsf{D}$ states that category C is a subcategory of category D. A category C is *small* if the collection C_{Mor}, and thus also the collection C_{Obj}, is a set. Cat is the category of all small categories. A category C is *locally small* if $\mathsf{C}(A, B)$ is a set for all objects A and B in C. Set denotes the category of all sets and all (total) maps. Cat and Set are not small but *locally small*.

A *(directed multi) graph* $\mathtt{G} = (\mathtt{G}_V, \mathtt{G}_E, sc^{\mathtt{G}}, tg^{\mathtt{G}})$ is given by a collection $\mathtt{G}_V$ of vertices, a collection $\mathtt{G}_E$ of edges and maps $sc^{\mathtt{G}} : \mathtt{G}_E \to \mathtt{G}_V$, $tg^{\mathtt{G}} : \mathtt{G}_E \to \mathtt{G}_V$ assigning to each edge its source and target vertex, respectively [9]. $\mathbf{0} = (\emptyset, \emptyset, id_\emptyset, id_\emptyset)$ is the *empty graph*. A graph $\mathtt{G}$ is *small* if $\mathtt{G}_V$ and $\mathtt{G}_E$ are sets. A *graph homomorphism* $\varphi : \mathtt{G} \to \mathtt{H}$ between two graphs $\mathtt{G} = (\mathtt{G}_V, \mathtt{G}_E, sc^{\mathtt{G}}, tg^{\mathtt{G}})$ and $\mathtt{H} = (\mathtt{H}_V, \mathtt{H}_E, sc^{\mathtt{H}}, tg^{\mathtt{H}})$ is a pair (φ_V, φ_E) of maps $\varphi_V : \mathtt{G}_V \to \mathtt{H}_V$, $\varphi_E : \mathtt{G}_E \to \mathtt{H}_E$ such that the following diagrams commute.

$$\begin{tikzcd} \mathtt{G}_E \arrow[r, "sc^{\mathtt{G}}"] \arrow[d, "\varphi_E"'] \arrow[dr, phantom, "\circlearrowleft"] & \mathtt{G}_V \arrow[d, "\varphi_V"] \\ \mathtt{H}_E \arrow[r, "sc^{\mathtt{H}}"'] & \mathtt{H}_V \end{tikzcd} \qquad \begin{tikzcd} \mathtt{G}_E \arrow[r, "tg^{\mathtt{G}}"] \arrow[d, "\varphi_E"'] \arrow[dr, phantom, "\circlearrowleft"] & \mathtt{G}_V \arrow[d, "\varphi_V"] \\ \mathtt{H}_E \arrow[r, "tg^{\mathtt{H}}"'] & \mathtt{H}_V \end{tikzcd}$$

The *identity graph homomorphism* $id_{\mathtt{G}}$ on a graph $\mathtt{G}$ is the pair $(id_{\mathtt{G}_V}, id_{\mathtt{G}_E})$ of identity maps and composition of graph homomorphisms $\varphi : \mathtt{G} \to \mathtt{H}$ and $\psi : \mathtt{H} \to \mathtt{K}$ is performed componentwise, i.e., $\varphi; \psi := (\varphi_V; \psi_V, \varphi_E; \psi_E)$. Graph is the category of all small graphs and all graph homomorphisms between them. The empty graph is the initial object in Graph. For convenience and uniformity, we will often consider a set A as a graph without edges.

The category comprising finite and small graphs as well as the underlying graphs of categories like Cat, Set, and Graph, for example, is denoted by GRAPH, while SET is the category containing all the corresponding collections of vertices and edges, respectively. Correspondingly, we denote the category with all small categories and categories like Cat, Set, and Graph as objects by CAT.

$gr(\mathsf{C})$ denotes the underlying graph of a category C, i.e., we have $\mathrm{gr}(\mathsf{C})_V := \mathsf{C}_{Obj}$ and $\mathrm{gr}(\mathsf{C})_E := \mathsf{C}_{Mor}$. Note that a functor $\mathbf{F} : \mathsf{C} \to \mathsf{D}$ is just a graph homomorphism $\mathbf{F} : gr(\mathsf{C}) \to gr(\mathsf{D})$ also preserving identities and composition. In other words, the assignments $\mathsf{C} \mapsto gr(\mathsf{C})$, $(\mathbf{F} : \mathsf{C} \to \mathsf{D}) \mapsto (\mathbf{F} : gr(\mathsf{C}) \to gr(\mathsf{D}))$ define a faithful forgetful functor $\mathbf{Gr} : \mathsf{Cat} \to \mathsf{Graph}$. It is well known that there is a functor $\mathbf{Pth} : \mathsf{Graph} \to \mathsf{Cat}$ left-adjoint to $\mathbf{Gr}$ assigning to any graph $\mathtt{G}$ the corresponding *path category* $\mathsf{P}(\mathtt{G})$.

In practical applications, it is often more convenient to work with interpretation categories instead of functor categories. An *interpretation* of a graph $\mathtt{G}$ in a category C, denoted by $\varphi : \mathtt{G} \to \mathsf{C}$, is a graph homomorphism φ from $\mathtt{G}$ to $gr(\mathsf{C})$. A *natural transformation* $\mu : \varphi \Rightarrow \psi$ between two interpretations $\varphi : \mathtt{G} \to \mathsf{C}$ and $\psi : \mathtt{G} \to \mathsf{C}$ is a family $\mu_v : \varphi_V(v) \to \psi_V(v)$, $v \in \mathtt{G}_V$ of morphism in C such that $\varphi_E(f); \mu_u = \mu_v; \psi_E(f)$ for all edges $f : v \to u$ in $\mathtt{G}$. All interpretations of $\mathtt{G}$ in C and all natural transformations between them constitute the *interpretation category* $[\mathtt{G} \to \mathsf{C}]$ with composition as the vertical composition of natural transformations. (In [10], interpretations $\varphi : \mathtt{G} \to \mathsf{C}$ are called "models of $\mathtt{G}$ in C" and the notation $\mathbf{Mod}(\mathtt{G}, \mathsf{C})$ is used instead of $[\mathtt{G} \to \mathsf{C}]$. For our purposes, the more neutral and general term "interpretation" is more convenient). For any categories C, D the assignments $(\mathbf{F} : \mathsf{C} \to \mathsf{D}) \mapsto (\mathbf{F} : gr(\mathsf{C}) \to gr(\mathsf{D}))$ define a full embedding of the traditional *functor category* $[\mathsf{C} \to \mathsf{D}]$ into the interpretation category $[gr(\mathsf{C}) \to \mathsf{D}]$.

Obviously, the category Graph is, by definition, isomorphic to the interpretation category $[\mathtt{MG} \to \mathsf{Set}]$ with $\mathtt{MG}$ as the graph $\begin{tikzcd} E \arrow[r, bend left, "sc"] \arrow[r, bend right, "tg"'] & V \end{tikzcd}$. On the other hand, the adjunction $\mathbf{Pth} \dashv \mathbf{Gr}$ ensures that for any small graph $\mathtt{G}$ the interpretation category $[\mathtt{G} \to \mathsf{C}]$ is isomorphic to the functor category $[\mathsf{P}(\mathtt{G}) \to \mathsf{C}]$ and thus a *presheaf topos*.

For any set I and any set A the set $A^I = \{\boldsymbol{a} : I \to A\}$ of all maps from I into A is a categorical product in Set with the family $(\pi_i : A^I \to A \mid i \in I)$ of projections defined by $\pi_i(\boldsymbol{a}) := \boldsymbol{a}(i)$ for all $\boldsymbol{a} \in A^I$. If I is finite with n elements (indices), A^I is therefore isomorphic to the n-ary Cartesian product A^n of A. If we equip a finite set $I_n = \{i_1, i_2, \ldots, i_n\}$ with a fixed total order $i_1 < i_2 < \ldots < i_n$, we can reuse the traditional tuple notation for elements in the Cartesian product A^n to also represent the elements in A^{I_n}: A map $\boldsymbol{a} : I_n \to A$ is represented by the tuple $(\boldsymbol{a}(i_1), \boldsymbol{a}(i_2), \ldots, \boldsymbol{a}(i_n))$. In this paper, we will often describe a map $\boldsymbol{a} : I_n \to A$ by simply declaring $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, i.e., $\boldsymbol{a}(i_j) = a_j$ for all $1 \leq j \leq n$. If $n = 0$, we have $I_0 = \emptyset$, and we will consequently represent the only map $\boldsymbol{a} : I_0 \to A$ by the *empty tuple* $()$. For any finite set A we denote its cardinality by $|A|$.

For any inclusion $A \subseteq B$ of sets, we denote by $\iota_{A,B} : A \hookrightarrow B$ the corresponding *inclusion map* with $\iota_{A,B}(a) = a$ for all $a \in A$. A graph $\mathtt{G}$ is a *subgraph* of a graph $\mathtt{H}$, $\mathtt{G} \sqsubseteq \mathtt{H}$ in symbols, if $\mathtt{G}_V \subseteq \mathtt{H}_V$, $\mathtt{G}_E \subseteq \mathtt{H}_E$ and the inclusion maps $\iota_{\mathtt{G}_V,\mathtt{H}_V} : \mathtt{G}_V \to \mathtt{H}_V$ and $\iota_{\mathtt{G}_E,\mathtt{H}_E} : \mathtt{G}_E \to \mathtt{H}_E$ establish a graph homomorphism $\iota_{\mathtt{G},\mathtt{H}} = (\iota_{\mathtt{G}_V,\mathtt{H}_V}, \iota_{\mathtt{G}_E,\mathtt{H}_E}) : \mathtt{G} \to \mathtt{H}$. $\iota_{\mathtt{G},\mathtt{H}}$ is also called an *inclusion graph homomorphism* or *graph inclusion*, for short.

A *partial map* $f : A \multimap B$ is given by a *domain of definition* $dom(f) \subseteq A$ and a total map from $dom(f)$ into B. The composition $f;g : A \multimap C$ of two partial maps $f : A \multimap B$ and $g : B \multimap C$ is defined by

- $dom(f;g) := \{x \in A \mid x \in dom(f), f(x) \in dom(g)\}$ and
- $f;g(a) := g(f(a))$ for all $a \in dom(f;g)$.

We consider any (total) map $f : A \to B$ as a partial map $f : A \multimap B$ with $dom(f) = A$.

3. Algebras and Term Algebras

Traditional expositions of Universal Algebra are based on finite Cartesian products. As a first step of a smooth transition from traditional algebras to graph algebras, we reformulate in this section some very basic concepts and results of Universal Algebra utilizing sets of maps A^I instead of finite Cartesian products A^n.

In parallel, we try to lift the traditional set-theoretic definitions, constructions and proofs in Universal Algebra to a more general and abstract level utilizing category-theoretic concepts and constructions. The objective is to pave the way from traditional operations and algebras via graph operations and graph algebras to operations and algebras in topoi.

3.1. Signatures, Algebras and Homomorphisms

To declare the arities of operation symbols, we use canonical finite indexing sets

$$I_0 = \emptyset, \quad I_n := \{i_1, i_2, \ldots, i_n\} \text{ for all } n \geq 1 \quad \text{and} \quad O = \{o\}. \tag{1}$$

For all $n \geq 2$, we assume I_n to be equipped with a fixed total order $i_1 < i_2 < \ldots < i_n$; thus, we can reuse the tuple notation to represent maps $\boldsymbol{a} : I_n \to A$ as discussed in Section 2.

Definition 3.1 (Signature). *A signature $\Sigma = (OP, ar)$ is given by*

- *A set OP of* operation symbols,
- *A map ar assigning to each operation symbol* op *as its* arity *a pair $ar(\mathsf{op}) = (I_{\mathsf{op}}, O_{\mathsf{op}})$ of finite sets with $I_{\mathsf{op}} = I_n$ for some $n \in \mathbb{N}$, and $O_{\mathsf{op}} = O = \{o\}$.*

*We say that $\mathsf{op} \in OP$ is an n-*ary operation symbol *if $I_{\mathsf{op}} = I_n$. If $I_{\mathsf{c}} = I_0$ for $\mathsf{c} \in OP$, and we also say that c is a* constant symbol.

Remark 3.1 (Sets as arities). *There can be arbitrarily and finitely many disconnected inputs for an algebraic operation. We have decided to work with explicit sets of names for the "input positions". In contrast to possibly multiple inputs, it is usually assumed that an algebraic operation has exactly one single output. For conformity reasons, we also introduce a name for the single output position. This brings us closer to graph algebras, where the single-output paradigm will be given up. The input as well as the output arity of a graph operation can be an arbitrary finite graph (see Definition 4.1).*

Definition 3.2 (Algebra). *Let Σ be a signature. A Σ-algebra $\mathcal{A} = (A, OP^{\mathcal{A}})$ is given by*

- *A set A, called the* carrier *of $\mathcal{A}$, and*
- *A family $OP^{\mathcal{A}} = (\mathsf{op}^{\mathcal{A}} \colon A^{I_{\mathsf{op}}} \to A^{O_{\mathsf{op}}} \mid \mathsf{op} \in OP)$ of maps called* operations.

*We say that $\mathsf{op}^{\mathcal{A}}$ is an n-*ary operation *if $I_{\mathsf{op}} = I_n$. If $I_{\mathsf{c}} = I_0$, we also say that $\mathsf{c}^{\mathcal{A}}$ is a* constant operation, *or simply a* constant.

In the case where the signature Σ has no constant symbols, the empty set constitutes a Σ-algebra, called the *empty Σ-algebra*.

Now, we reformulate the traditional concept of homomorphism.

Definition 3.3 (Homomorphism). *Let Σ be a signature. A* Σ-homomorphism *$h : \mathcal{A} \to \mathcal{B}$ between two Σ-algebras $\mathcal{A}$ and $\mathcal{B}$ is a map $h : A \to B$ satisfying the following homomorphism condition*

$$\textbf{(HC)} \quad \mathsf{op}^{\mathcal{A}}(\boldsymbol{a}); h = \mathsf{op}^{\mathcal{B}}(\boldsymbol{a}; h) \quad \textit{for all } \mathsf{op} \in OP \textit{ and all } \boldsymbol{a} \in A^{I_{\mathsf{op}}}.$$

$$\begin{tikzcd}
I_{\mathsf{op}} \arrow[r, "\boldsymbol{a}"] \arrow[dr, "\boldsymbol{a};h"'] & A \arrow[d, "h"] \arrow[dl, phantom, "\circlearrowleft"] \arrow[dr, phantom, "\circlearrowleft"] & O_{\mathsf{op}} \arrow[l, "\mathsf{op}^{\mathcal{A}}(\boldsymbol{a})"'] \arrow[dl, "\mathsf{op}^{\mathcal{B}}(\boldsymbol{a};h)"] \\
 & B &
\end{tikzcd}
\qquad
\begin{tikzcd}
A^{I_{\mathsf{op}}} \arrow[r, "\mathsf{op}^{\mathcal{A}}"] \arrow[d, "\_;h"'] \arrow[dr, phantom, "\circlearrowleft"] & A^{O_{\mathsf{op}}} \arrow[d, "\_;h"] \\
B^{I_{\mathsf{op}}} \arrow[r, "\mathsf{op}^{\mathcal{B}}"'] & B^{O_{\mathsf{op}}}
\end{tikzcd}$$

For any sets A, B and I each map $h : A \to B$ induces a map $\_;h : A^I \to B^I$ and thus we can, more abstractly but equivalently, express the homomorphism condition **(HC)** with the requirement that the above right square of maps commutes. Note that, in the case of constant symbols $\mathsf{c} \in OP$ with $I_{\mathsf{c}} = I_0$, the homomorphism condition turns into the equation $\mathsf{op}^{\mathcal{A}}();h = \mathsf{op}^{\mathcal{B}}()$ if we apply our conventions in Section 2 concerning the tuple notation.

Given any Σ-algebra $\mathcal{A}$, the identity map on the carrier set $id_A : A \to A$ induces an identity Σ-homomorphism $id_{\mathcal{A}} : \mathcal{A} \to \mathcal{A}$. Similarly, given any Σ-homomorphisms $f : \mathcal{A} \to \mathcal{B}$, $g : \mathcal{B} \to \mathcal{C}$, the composition $f;g : A \to C$ of the underlying maps induces a Σ-homomorphism $f;g : \mathcal{A} \to \mathcal{C}$. This defines the category $\mathsf{Alg}(\Sigma)$ with all Σ-algebras as objects and all Σ-homomorphisms as morphisms.

Proposition 3.1 (Forgetful Functor)**.** *The assignments $\mathcal{A} \mapsto A$ for Σ-algebras and $(f : \mathcal{A} \to \mathcal{B}) \mapsto (f : A \to B)$ for Σ-homomorphisms define a faithful forgetful functor*

$$\mathbf{U}_\Sigma : \mathsf{Alg}(\Sigma) \to \mathsf{Set}. \tag{2}$$

A characteristic for any incarnation of the concept *algebra* is that the corresponding categories of algebras inherit all limits from the respective *underlying category*. For the abstract concept of **F**-*algebra* for an arbitrary functor $\mathbf{F}\colon \mathsf{C} \to \mathsf{C}$, for example, the category of all **F**-algebras inherits all limits from the category C [11].

It is well-known that the category $\mathsf{Alg}(\Sigma)$ inherits all limits from the category Set. Following our methodological intention to lift things up to a more categorical element-free level, we demonstrate that the decision to work with sets of maps A^I instead of Cartesian products A^n enables us to give a pure categorical concise proof of this classical result. We intend to carry out all later constructions and argumentations within Set, and thus we restrict ourselves to small limits.

Theorem 3.1 (Limits)**.** $\mathsf{Alg}(\Sigma)$ *inherits any small limit from the category* Set*, i.e., the functor* $\mathbf{U}_\Sigma : \mathsf{Alg}(\Sigma) \to \mathsf{Set}$ *creates small limits.* $\mathsf{Alg}(\Sigma)$ *has therefore all small limits since* Set *does.*

Proof. Let J be a small graph and $\delta : \mathsf{J} \to \mathsf{Alg}(\Sigma)$ be a diagram in $\mathsf{Alg}(\Sigma)$ where $\delta_v = \mathcal{A}_v = (A_v, OP^{\mathcal{A}_v})$ for all vertices v in J_V. We have to show that any limit cone $\pi : L \Rightarrow \delta;\mathbf{U}_\Sigma$ over the translated diagram $\delta;\mathbf{U}_\Sigma : \mathsf{J} \to \mathsf{Set}$ induces a limit cone $\pi : \mathcal{L} \Rightarrow \delta$ over δ in $\mathsf{Alg}(\Sigma)$ such that $\mathcal{L}$ is a Σ-algebra with L as its carrier.

To define the operations in $\mathcal{L}$, we note that for any operation symbol op in OP and any map $\boldsymbol{l} : I_{\mathsf{op}} \to L$ we obtain a commutative cone $\boldsymbol{l};\pi : I_{\mathsf{op}} \Rightarrow \delta;\mathbf{U}_\Sigma$ in Set with $(\boldsymbol{l};\pi)_v := \boldsymbol{l};\pi_v : I_{\mathsf{op}} \to A_v$ for all v in J_V. Applying the respective operations $\mathsf{op}^{\mathcal{A}_v}$ to the maps $\boldsymbol{l};\pi_v$ gives us a new cone $\mathsf{op}^\delta(\boldsymbol{l};\pi) : O_{\mathsf{op}} \Rightarrow \delta;\mathbf{U}_\Sigma$ in Set with $\mathsf{op}^\delta(\boldsymbol{l};\pi)_v := \mathsf{op}^{\mathcal{A}_v}(\boldsymbol{l};\pi_v) : O_{\mathsf{op}} \to A_v$ for all vertices v in J_V. Now, for any edge $e : v \to w$ in J_E, we have that $\delta_e : \delta_v \to \delta_w$ is a Σ-homomorphism from $\mathcal{A}_v$ to $\mathcal{A}_w$ and thus, by the homomorphism condition and commutativity of $\pi : L \Rightarrow \delta;\mathbf{U}_\Sigma$, we have

$$\mathsf{op}^{\mathcal{A}_v}(\boldsymbol{l};\pi_v);\delta_e = \mathsf{op}^{\mathcal{A}_w}(\boldsymbol{l};\pi_v;\delta_e) = \mathsf{op}^{\mathcal{A}_w}(\boldsymbol{l};\pi_w) \tag{3}$$

which encapsulates that the cone $\mathsf{op}^{\delta}(\boldsymbol{l};\pi) : O_{\mathsf{op}} \Rightarrow \delta;\mathbf{U}_{\Sigma}$ is commutative. By the universal property of π, there is a unique map $k : O_{\mathsf{op}} \to L$ which satisfies

$$k;\pi_v = \mathsf{op}^{\mathcal{A}_v}(\boldsymbol{l};\pi_v) \tag{4}$$

for all v in J_V. By defining $\mathsf{op}^{\mathcal{L}}(\boldsymbol{l}) := k$, we ensure that each map $\pi_v : L \to A_v$ induces a Σ-homomorphism $\pi_v : \mathcal{L} \to \mathcal{A}_v$ for all v in J_v. Thus, we indeed obtain a commutative cone $\pi : \mathcal{L} \Rightarrow \delta$ in $\mathsf{Alg}(\Sigma)$.

It remains to show that $\pi : \mathcal{L} \Rightarrow \delta$ is a limit cone, i.e., for any other commutative cone $p : \mathcal{X} \Rightarrow \delta$ in $\mathsf{Alg}(\Sigma)$, we have to show that there is a Σ-homomorphism $\kappa : \mathcal{X} \to \mathcal{L}$ such that $\kappa;\pi_v = p_v$ for all v in J_V. Note that p induces a commutative cone $p;\mathbf{U}_{\Sigma} : X \Rightarrow \delta;\mathbf{U}_{\Sigma}$ in Set with $(p;\mathbf{U}_{\Sigma})_v := p_v : X \to A_v$ for all v in J_V. As π is a limit cone over $\delta;\mathbf{U}_{\Sigma}$, there exists a unique map $\kappa : X \to L$ such that $\kappa;\pi_v = p_v$ for all v in J_V. We claim that κ extends to the desired Σ-homomorphism by showing that

$$\mathsf{op}^{\mathcal{X}}(\boldsymbol{x});\kappa = \mathsf{op}^{\mathcal{L}}(\boldsymbol{x};\kappa) \tag{5}$$

for any op in OP and $\boldsymbol{x} : I_{\mathsf{op}} \to X$. By definition, $\mathsf{op}^{\mathcal{L}}(\boldsymbol{x};\kappa)$ is the unique map such that $\mathsf{op}^{\mathcal{L}}(\boldsymbol{x};\kappa);\pi_v = \mathsf{op}^{\mathcal{A}_v}(\boldsymbol{x};\kappa;\pi_v)$ holds for all v in J_V. Indeed, the map $\mathsf{op}^{\mathcal{X}}(\boldsymbol{x});\kappa : O_{\mathsf{op}} \to X$ also satisfies this equality for all v in J_V as

$$\mathsf{op}^{\mathcal{X}}(\boldsymbol{x});\kappa;\pi_v = \mathsf{op}^{\mathcal{X}}(\boldsymbol{x});p_v = \mathsf{op}^{\mathcal{A}_v}(\boldsymbol{x};p_v) = \mathsf{op}^{\mathcal{A}_v}(\boldsymbol{x};\kappa;\pi_v).$$

By the uniqueness of mediating morphisms. we obtain (5), which shows that $\kappa : X \to L$ is indeed a Σ-homomorphism from $\kappa : \mathcal{X} \to \mathcal{L}$. Moreover, it is the unique homomorphism such that $\kappa;\pi_v = p_v$ and thus, $\pi : \mathcal{L} \Rightarrow \delta$ is a limit cone. □

Remark 3.2 (Hom-sets)**.** *The proof of Theorem 3.1 is based on the convention in Section 2 that we consider A^I as a shorthand notation for the collection (hom-set)* $\mathsf{Set}(I, A)$ *of all morphisms in* Set *from I to A.* $\mathsf{Set}(I, A)$ *is a set since* Set *is* locally small*.* $\mathsf{Set}(I, A)$ *is isomorphic to a corresponding* exponential object *in* Set*, but this isomorphism does not play any role in this paper.*

3.2. Subalgebras

As in the traditional approach, we can define subalgebras by means of set inclusions.

Definition 3.4 (Subalgebras)**.** *Let $\Sigma = (OP, ar)$ be a signature. A Σ-algebra $\mathcal{A} = (A, OP^{\mathcal{A}})$ is a* Σ-subalgebra *of a Σ-algebra $\mathcal{B} = (B, OP^{\mathcal{B}})$, $\mathcal{A} \sqsubseteq \mathcal{B}$ in symbols, if $A \subseteq B$ and for all $\mathsf{op} \in OP$ and $\boldsymbol{a} : I_{\mathsf{op}} \to A$ the following diagram commutes:*

$$\begin{tikzcd}
I_{\mathsf{op}} \arrow[r, "\boldsymbol{a}"] \arrow[dr, "\boldsymbol{a};\iota_{A,B}"'] & A \arrow[d, hook, "\iota_{A,B}"] \arrow[dl, phantom, "\circlearrowleft"] \arrow[dr, phantom, "\circlearrowleft"] & O_{\mathsf{op}} \arrow[l, "\mathsf{op}^{\mathcal{A}}(\boldsymbol{a})"'] \arrow[dl, "\mathsf{op}^{\mathcal{B}}(\boldsymbol{a};\iota_{A,B})"] \\
 & B &
\end{tikzcd}$$

Here, $\iota_{A,B} : A \hookrightarrow B$ is the corresponding inclusion map from A into B.

A comparison of Definitions 3.3 and 3.4 makes it, however, obvious that we also can simply describe Σ-subalgebras as special kinds of Σ-homomorphisms.

Corollary 3.1 (Subalgebras as Inclusion Homomorphisms)**.** *For Σ-algebras $\mathcal{A}$ and $\mathcal{B}$ such that $A \subseteq B$, we have that $\mathcal{A}$ is a Σ-subalgebra of $\mathcal{B}$ if, and only if, the inclusion map $\iota_{A,B} : A \hookrightarrow B$ establishes a Σ-homomorphism $\iota_{A,B} : \mathcal{A} \hookrightarrow \mathcal{B}$.*

Since one of our objectives is to lift the traditional set-theoretic exposition of Universal Algebra to a more category-theoretic one, we will use, from now on, the concepts "Σ-subalgebra" and "inclusion Σ-homomorphism" interchangeably.

We know that the monomorphisms (epimorphisms) in Set are exactly the injective (surjective) maps, respectively. Any faithful functor reflects monomorphisms and epimorphisms. The forgetful functor $\mathbf{U}_\Sigma : \mathsf{Alg}(\Sigma) \to \mathsf{Set}$ is faithful, and thus, we obtain

Corollary 3.2 (Injective and surjective Homomorphisms)**.** *If the underlying map $f\colon A \to B$ of a Σ-homomorphism $f : \mathcal{A} \to \mathcal{B}$ is injective (surjective), then $f : \mathcal{A} \to \mathcal{B}$ is a monomorphism (epimorphism) in* $\mathsf{Alg}(\Sigma)$.

In the traditional set-theoretic approach to Universal Algebra, a preferred tool to describe, construct and reason about subalgebras are subsets of the carrier which are closed with respect to applications of the operations in the algebra.

Definition 3.5 (Closedness)**.** *Let $\mathcal{B} = (B, OP^{\mathcal{B}})$ be a Σ-algebra. We say a subset $A \subseteq B$ is* closed in $\mathcal{B}$ *if for all $\mathtt{op} \in OP$ and $\boldsymbol{a} : I_{\mathtt{op}} \to A$ the result $\mathtt{op}^{\mathcal{B}}(\boldsymbol{a};\iota_{A,B}) : O_{\mathtt{op}} \to B$ of applying the operation $\mathtt{op}^{\mathcal{B}}$ in $\mathcal{B}$ to the input $\boldsymbol{a};\iota_{A,B} : I_{\mathtt{op}} \to B$ factors through the inclusion map $\iota_{A,B} : A \hookrightarrow B$, i.e., there exists a map $r_{\boldsymbol{a}} : O_{\mathtt{op}} \to B$ such that the following diagram commutes:*

$$\begin{tikzcd}
I_{\mathtt{op}} \arrow[r, "\boldsymbol{a}"] \arrow[dr, "\boldsymbol{a};\iota_{A,B}"'] & A \arrow[d, hook, "\iota_{A,B}"] & O_{\mathtt{op}} \arrow[l, dashed, "r_{\boldsymbol{a}}"'] \arrow[dl, "\mathtt{op}^{\mathcal{B}}(\boldsymbol{a};\iota_{A,B})"] \\
 & B &
\end{tikzcd}$$

$\mathcal{A}$ is a Σ-subalgebra of $\mathcal{B}$ then the carrier A of $\mathcal{A}$ is obviously closed with respect to all the operations in $\mathcal{B}$. On the other hand, the inclusion map $\iota_{A,B} : A \hookrightarrow B$ is a monomorphism in Set. Therefore the map $r_{\boldsymbol{a}} : O_{\mathtt{op}} \to B$ in Definition 3.5 is unique if it exists. In such a way, the assignments $\boldsymbol{a} \mapsto r_{\boldsymbol{a}}$ define a total operation from $A^{I_{\mathtt{op}}}$ to $A^{O_{\mathtt{op}}}$ if A is closed, and we obtain the following result.

Proposition 3.2 (Subalgebra $\cong$ Closed Subset)**.** *There is a one-to-one correspondence between Σ-subalgebras of $\mathcal{B}$ and closed subsets of $\mathcal{B}$.*

Proposition 3.2 suggests that there may actually be no need for the auxiliary concept *closed subset* in a more category-theoretic approach. This conjecture is supported by the observation that we can reconstruct the standard result that closed subsets are closed with respect to intersection, reformulated in terms of inclusion homomorphisms, as a special case of Theorem 3.1. To see this, we have to realize that the intersection of subsets can be described as a special limit construction, namely *multiple pullbacks*, in Set.

Remark 3.3 (Multiple Pullbacks)**.** *Let I be a set and M be an I-indexed family $(A_i \subseteq B \mid i \in I)$ of subsets of a set B. We can describe this situation witha diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Set}$ with $\mathtt{MP}(I)$ as a small graph given by $\mathtt{MP(I)}_V := I \cup \{*\}$, $\mathtt{MP(I)}_E := \{e_i : i \to * \mid i \in I\}$, and δ defined by $\delta_i := A_i$ for all $i \in I$, $\delta_* := B$ and inclusion maps $\delta_{e_i} := \iota_{A_i,B} : A_i \hookrightarrow B$ for all $i \in I$.*

It is well known and straightforward to prove that the intersection $\cap M := \bigcap_{i\in I} A_i$ together with the inclusion maps $\iota_{\cap M,A_i} : \cap M \hookrightarrow A_i$, $i \in I$ and $\iota_{\cap M,B} = \iota_{\cap M,A_i};\iota_{A_i,B} : \cap M \hookrightarrow B$ is a limit cone of the diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Set}$ in Set.

Limits of this shape are also called multiple pullbacks *and they reflect monomorphisms: For any category C, any diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{C}$ and any limit cone $p\colon L \Rightarrow \delta$, all the morphisms $p_i\colon L \to \delta_i$, $i \in I$ are monomorphisms in C as as long as all the morphisms $\delta_{e_i}\colon \delta_i \to \delta_*$, $i \in I$ exist.*

Due to Remark 3.3, we can now replace and enhance the traditional statement

"If M is a family of closed subsets in $\mathcal{B}$, then its intersection $\cap M$ is closed as well"

by the following corollary of Theorem 3.1.

Corollary 3.3 (Intersection of Subalgebras). *For any set I, any Σ-algebra $\mathcal{B}$, and any diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Alg}(\Sigma)$ of Σ-subalgebras $\delta_{e_i} = \iota_{A_i,B} : \mathcal{A}_i \hookrightarrow \mathcal{B}$, $i \in I$ of $\mathcal{B}$ there is a unique Σ-subalgebra $\mathcal{L} = (L, OP^{\mathcal{L}})$ of $\mathcal{B}$ with $L = \bigcap_{i\in I} A_i$ that is a Σ-subalgebra of $\mathcal{A}_i$ for all $i \in I$.*

Moreover, the inclusion Σ-homomorphisms $\iota_{L,A_i} : \mathcal{L} \hookrightarrow \mathcal{A}_i$, $i \in I$ and $\iota_{L,B} : \mathcal{L} \hookrightarrow \mathcal{B}$ constitutes a multiple pullback, i.e., a limit cone of the diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Alg}(\Sigma)$ in $\mathsf{Alg}(\Sigma)$.

We also call $\mathcal{L} = (L, OP^{\mathcal{L}})$ the intersection *of the I-indexed family $\mathcal{M} = (\mathcal{A}_i \mid i \in I)$ of Σ-subalgebras of $\mathcal{B}$ and may use the notations $\bigcap \mathcal{M}$, $\bigcap_{i\in I} \mathcal{A}_i$ or, simply, $\bigcap \mathcal{A}_i$ to denote $\mathcal{L}$.*

Traditionally, the Σ-subalgebra $\mathcal{R}(A, \mathcal{B})$ of a Σ-algebra $\mathcal{B}$ *generated by a subset $A \subseteq B$* can be defined as the Σ-subalgebra with the carrier $R(A, \mathcal{B})$ constructed as the intersection of the following family of closed sets in $\mathcal{B}$:

$$R(A, \mathcal{B}) := \cap\{X \subseteq B \mid X \text{ closed in } \mathcal{B} \text{ and } A \subseteq X\}. \tag{6}$$

Since the collection of all subsets of a set B is a set as well, this definition matches the pattern of Corollary 3.3. We only have to choose for I the set $\{X \subseteq B \mid X \text{ closed in } \mathcal{B} \text{ and } A \subseteq X\}$ itself or any isomorphic set.

Using this sleight of hand, we can take full advantage of the universal property of the intersection of subalgebras in $\mathsf{Alg}(\Sigma)$, as stated in Corollary 3.3, and lift up the concept "generated by a subset" to the concept "accessible via a map".

Definition 3.6 (Subalgebra accessible via a Map). *For any Σ-algebra $\mathcal{B}$ and any map $f\colon A \to B$, let $\mathcal{M}$ be the set of all Σ-subalgebras $\mathcal{X}$ of $\mathcal{B}$ such that f factors through the inclusion map $\iota_{X,B} : X \hookrightarrow B$, i.e., there exists a map $f_{\mathcal{X}}\colon A \to X$ such that $f_{\mathcal{X}} ; \iota_{X,B} = f$.*

We denote by $\mathcal{R}(f, \mathcal{B})$ the intersection of $\mathcal{M}$, according to Corollary 3.3. In particular, the carrier of $\mathcal{R}(f, \mathcal{B})$ is the intersection $R(f, \mathcal{B}) := \bigcap\{X \mid \mathcal{X} \in \mathcal{M}\}$ of sets. We call $\mathcal{R}(f, \mathcal{B})$ the Σ-subalgebra of $\mathcal{B}$ accessible (reachable) via f *or the* homomorphic image of A with respect to f.

In the case of inclusion maps $f = \iota_{A,B}\colon A \hookrightarrow B$, we also use the traditional notation $\mathcal{R}(A, \mathcal{B})$ instead of $\mathcal{R}(\iota_{A,B}, \mathcal{B})$ and also call $\mathcal{R}(A, \mathcal{B})$ the Σ-subalgebra of $\mathcal{B}$ generated by A.

Note that the map $f_{\mathcal{X}}\colon A \to X$ in Definition 3.6 is unique if it exists, since the inclusion map $\iota_{X,B} : X \hookrightarrow B$ is a monomorphism in Set.

Corollary 3.4 (Homomorphic image includes Image). *For any Σ-algebra $\mathcal{B}$ and any map $f\colon A \to B$, we have $f(A) \subseteq R(f, \mathcal{B})$ for the* (set-theoretic) image *$f(A) := \{f(a) \mid a \in A\}$ of A with respect to the map f.*

Proof. This follows immediately from the observation that $f(A) = \bigcap\{Y \mid Y \in \mathcal{N}\}$ for the set $\mathcal{N}$ of all subsets Y of B such that f factors through the inclusion map $\iota_{Y,B} : Y \hookrightarrow B$ and that $\{X \mid \mathcal{X} \in \mathcal{M}\} \subseteq \mathcal{N}$ due to the definition of $\mathcal{M}$ and $\mathcal{R}(f, \mathcal{B})$ in Definition 3.6 and the definition of $\mathcal{N}$. □

Remark 3.4 (Well-powered). *In category theory, the adjective* well-powered *is used for categories C where for all objects A in C the collection of all subobjects of A is a set.*

Of course, we do have the traditional concept of generated algebra and corresponding results available.

Definition 3.7 (Accessible and Generated Algebras)**.** *Let $\mathcal{B}$ be a Σ-algebra.*

1. *$\mathcal{B}$ is* accessible via a map *$f : A \to B$ if $\mathcal{R}(f,\mathcal{B}) = \mathcal{B}$.*
2. *If $\mathcal{B}$ is accessible via an inclusion map $\iota_{A,B} : A \to B$, i.e., if $\mathcal{R}(A,\mathcal{B}) = \mathcal{R}(\iota_{A,B},\mathcal{B}) = \mathcal{B}$, we also say that $\mathcal{B}$ is* generated by *A.*
3. *$\mathcal{B}$ is said to be* generated *if it is generated by the empty set, i.e., accessible via the unique map $\iota_{\emptyset,B} : \emptyset \hookrightarrow B$ from the initial object $\emptyset$ in* Set *to B.*

Corollary 3.5. *A Σ-algebra $\mathcal{B}$ is generated if, and only if, there are no proper Σ-subalgebras of $\mathcal{B}$.*

Corollary 3.6. *If a signature Σ has no constant symbols, then the empty Σ-algebra is the only generated Σ-algebra.*

The concept *accessible via a map* can be utilized to find a characterization of epimorphisms in $\mathsf{Alg}(\Sigma)$. First, we observe that "accessible via a map" implies "epic".

Lemma 3.1 (Accessible implies Epic)**.** *A Σ-homomorphism $f\colon \mathcal{A} \to \mathcal{B}$ is an epimorphism in* $\mathsf{Alg}(\Sigma)$ *if $\mathcal{B}$ is accessible via the underlying map $f\colon A \to B$, i.e., if $\mathcal{B} = \mathcal{R}(f,\mathcal{B})$.*

Proof. We consider arbitrary Σ-homomorphisms $g,h : \mathcal{B} \to \mathcal{C}$ such that $f;g = f;h$.

We know that the set $X = \{b \in B \mid g(b) = h(b)\} \subseteq B$ together with the inclusion map $\iota_{X,B} : X \hookrightarrow B$ is an equalizer of the maps $g,h : B \to C$ in Set. According to Theorem 3.1, there is a unique Σ-algebra $\mathcal{X} = (X, OP^{\mathcal{X}})$ such that $\iota_{X,B} : X \hookrightarrow B$ becomes an inclusion Σ-homomorphism $\iota_{X,B} : \mathcal{X} \hookrightarrow \mathcal{B}$, which is, moreover, the equalizer of the Σ-homomorphisms $g,h : \mathcal{B} \to \mathcal{C}$. Assumption $f;g = f;h$ ensures that there exists a unique map $f_{\mathcal{X}}\colon A \to X$ with $f_{\mathcal{X}}\,;\iota_{X,B} = f$. Due to the construction of $\mathcal{R}(f,\mathcal{B})$ in Definition 3.6, we have an inclusion Σ-homomorphism $\iota_{R(f,\mathcal{B}),X} : \mathcal{R}(f,\mathcal{B}) \hookrightarrow \mathcal{X}$.

Accessibility of $\mathcal{B}$ means $\mathcal{B} = \mathcal{R}(f,\mathcal{B})$, and thus we obtain, finally, $\mathcal{X} = \mathcal{B}$. This means, however, that the equalizer of g and h is the identity on B thus, we have $g = h$ as required. □

To show that, on the other hand, epic implies accessible, we can take advantage of the following result.

Proposition 3.3 (Subalgebras are Regular Monos)**.** *For any Σ-subalgebra $\iota_{\mathcal{A},\mathcal{B}}\colon \mathcal{A} \hookrightarrow \mathcal{B}$ of a Σ-algebra $\mathcal{B}$ there exists a Σ-algebra $\mathcal{C}$ and parallel Σ-homomorphisms $g,h\colon \mathcal{B} \to \mathcal{C}$ such that $\iota_{\mathcal{A},\mathcal{B}}\colon \mathcal{A} \hookrightarrow B$ is the equalizer of g and h.*

Proof. We construct the pushout of the span $B \xhookleftarrow{\iota_{A,B}} A \xhookrightarrow{\iota_{A,B}} B$ of inclusion maps (see the left diagram below). We set $C := B +_A B$, $g := \kappa_1$, and $h := \kappa_2$. Since $\iota_{A,B}$ is injective, both maps $\kappa_1, \kappa_2\colon B \to B +_A B$ are injective too, and, moreover, the pushout square is as well a pullback square. This ensures, especially, that $\iota_{A,B} : A \hookrightarrow B$ is the equalizer of the maps $\kappa_1, \kappa_2 : B \to B +_A B$ in Set. The pushout property of the square provides a unique map $k : B +_A B \to B$ such that

$$\kappa_1; k = \kappa_2; k = id_B \tag{7}$$

$$\begin{tikzcd}
& A \arrow[ld, hook', "\iota_{A,B}"'] \arrow[rd, hook, "\iota_{A,B}"] & \\
B \arrow[rd, "\kappa_1"] \arrow[rdd, bend right, "id_B"'] & po & B \arrow[ld, "\kappa_2"'] \arrow[ldd, bend left, "id_B"] \\
\circlearrowleft & B +_A B \arrow[d, dashed, "k"] & \circlearrowleft \\
& B &
\end{tikzcd}
\qquad
\begin{tikzcd}
& I_{\mathsf{op}} \arrow[ldd, bend right, "\boldsymbol{b}_1"'] \arrow[d, dashed, "\boldsymbol{a}"'] \arrow[dd, bend left, "\boldsymbol{c}"] \arrow[rdd, bend left, "\boldsymbol{b}_2"] & \\
& A \arrow[ld, hook', "\iota_{A,B}"'] \arrow[rd, hook, "\iota_{A,B}"] & \\
B \arrow[rd, "\kappa_1"] \arrow[rdd, bend right, "id_B"'] & pb & B \arrow[ld, "\kappa_2"'] \arrow[ldd, bend left, "id_B"] \\
\circlearrowleft & B +_A B \arrow[d, dashed, "k"] & \circlearrowleft \\
& B &
\end{tikzcd}$$

Operations on C: We now extend C to a Σ-algebra $\mathcal{C} = (C, OP^{\mathcal{C}})$ by defining for each op in OP a corresponding operation $\mathsf{op}^{\mathcal{C}} \colon (B +_A B)^{I_{\mathsf{op}}} \to (B +_A B)^{O_{\mathsf{op}}}$. For any $\boldsymbol{c} : I_{\mathsf{op}} \to B +_A B$ in $(B +_A B)^{I_{\mathsf{op}}}$, we do have four possible cases.

Case 1 $\boldsymbol{c}$ factors through κ_1: There exists a map $\boldsymbol{b}_1 : I_{\mathsf{op}} \to B$ such that $\boldsymbol{b}_1; \kappa_1 = \boldsymbol{c}$ (see the right diagram above). $\boldsymbol{b}_1$ is unique since κ_1 is a monomorphism. We simply set

$$\mathsf{op}^{\mathcal{C}}(\boldsymbol{c}) = \mathsf{op}^{\mathcal{C}}(\boldsymbol{b}_1; \kappa_1) := \mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_1); \kappa_1 \tag{8}$$

$$\begin{tikzcd}
I_{\mathsf{op}} \arrow[r, "\boldsymbol{b}_1"] \arrow[rd, "\boldsymbol{c}"'] & B \arrow[d, "\kappa_1"] & O_{\mathsf{op}} \arrow[l, "\mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_1)"'] \arrow[ld, "\mathsf{op}^{\mathcal{C}}(\boldsymbol{c})"] \\
\circlearrowleft & B +_A B & \circlearrowleft
\end{tikzcd}$$

Case 2 $\boldsymbol{c}$ factors through κ_2: Analogous to Case 1.

Case 3 Overlapping of Case 1 and 2: There exist maps $\boldsymbol{b}_1, \boldsymbol{b}_2 : I_{\mathsf{op}} \to B$ such that $\boldsymbol{b}_1; \kappa_1 = \boldsymbol{c} = \boldsymbol{b}_2; \kappa_1$. Due to the pullback property of the square, there exists a unique $\boldsymbol{a} : I_{\mathsf{op}} \to A$ such that $\boldsymbol{b}_1 = \boldsymbol{a}; \iota_{A,B} = \boldsymbol{b}_2$. The homomorphism property of $\iota_{\mathcal{A},\mathcal{B}}$ ensures $\mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_1) = \mathsf{op}^{\mathcal{B}}(\boldsymbol{a}; \iota_{A,B}) = \mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_2) = \mathsf{op}^{\mathcal{A}}(\boldsymbol{a}); \iota_{A,B}$ and we obtain, finally, $\mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_1); \kappa_1 = \mathsf{op}^{\mathcal{A}}(\boldsymbol{a}); \iota_{A,B}; \kappa_1 = \mathsf{op}^{\mathcal{A}}(\boldsymbol{a}); \iota_{A,B}; \kappa_2 = \mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_2); \kappa_2$. That is, in the event of an overlapping, Case 1 and Case 2 define the same output $\mathsf{op}^{\mathcal{C}}(\boldsymbol{c}) = \mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_1); \kappa_1 = \mathsf{op}^{\mathcal{B}}(\boldsymbol{b}_2); \kappa_2$. Note that there will always be an overlapping for constant symbols!

Case 4 $\boldsymbol{c}$ factors neither through κ_1 nor through κ_2: This can only happen if $I_{\mathsf{op}} = I_n$ with $n \geq 2$. Utilizing the operations in $\mathcal{B}$ to a maximal extent, Cases 1 and 2 define a partial map from $(B +_A B)^{I_{\mathsf{op}}}$ to $(B +_A B)^{O_{\mathsf{op}}}$. However, since we restrict ourselves to total operations, we have to find an ad hoc *totalization trick* to turn $\mathsf{op}^{\mathcal{C}}$ into a total operation. Employing (7), we may decide to utilize the operations in $\mathcal{B}$ to produce outputs in the left copy of B in $B +_A B$. We set

$$\mathsf{op}^{\mathcal{C}}(\boldsymbol{c}) := \mathsf{op}^{\mathcal{B}}(\boldsymbol{c}; k); \kappa_1 \tag{9}$$

$$\begin{tikzcd}
I_{\mathrm{op}} \arrow[r, "\boldsymbol{c};k"] \arrow[dr, "\boldsymbol{c}"'] \arrow[r, phantom, bend right=60, "\circlearrowleft"] & B \arrow[d, shift right, "\kappa_1"'] & O_{\mathrm{op}} \arrow[l, "\mathrm{op}^{\mathcal{B}}(\boldsymbol{c};k)"'] \arrow[dl, "\mathrm{op}^{\mathcal{C}}(\boldsymbol{c})"] \arrow[l, phantom, bend left=60, "\circlearrowleft"] \\
 & B +_A B \arrow[u, shift right, "k"'] &
\end{tikzcd}$$

The operations in $\mathcal{C}$ are defined by (8) exactly in a way that the maps κ_1 and κ_2 become Σ-homomorphisms $\kappa_1, \kappa_2 : \mathcal{B} \to \mathcal{C}$. Note that **Case 4** has no relevance for the homomorphism property of κ_1 and κ_2! Theorem 3.1 ensures, finally, that the inclusion Σ-homomorphism $\iota_{\mathcal{A},\mathcal{B}} \colon \mathcal{A} \to \mathcal{B}$ is the equalizer of the Σ-homomorphisms $\kappa_1, \kappa_2 : \mathcal{B} \to \mathcal{C}$. □

Regularity of $\mathsf{Alg}(\Sigma)$ entails that the concepts *accessible* and *epic* are equivalent.

Proposition 3.4 (Accessible $\cong$ Epic)**.** *For any Σ-homomorphism $f \colon \mathcal{A} \to \mathcal{B}$, it holds that $\mathcal{B}$ is accessible via the underlying map $f : A \to B$; i.e., in other words, $\mathcal{B}$ is equal to the homomorphic image $\mathcal{R}(f, \mathcal{B})$ of $\mathcal{A}$ with respect to f, if, and only if, $f \colon \mathcal{A} \to \mathcal{B}$ is an epimorphism in* $\mathsf{Alg}(\Sigma)$.

Proof. "Accessible implies epic" has been shown in Lemma 3.1. We now show that "epic implies accessible": We consider an arbitrary Σ-subalgebra $\mathcal{X}$ of $\mathcal{B}$ such that there exists a map $f_{\mathcal{X}} \colon A \to X$ with $f_{\mathcal{X}} ; \iota_{X,B} = f$. Due to Proposition 3.3, there exist Σ-homomorphisms $g, h \colon \mathcal{B} \to \mathcal{C}$ such that $\iota_{\mathcal{X},\mathcal{B}} \colon \mathcal{X} \to \mathcal{B}$ is the equalizer of g and h. Due to the assumption $f_{\mathcal{X}} ; \iota_{X,B} = f$, we obtain $f;g = f;h$ and thus $g = h$ since f is epic. This means, however, $\mathcal{X} = \mathcal{B}$ and, finally, $\mathcal{R}(f, \mathcal{B}) = \mathcal{B}$ due to the construction of $\mathcal{R}(f, \mathcal{B})$ in Definition 3.6. □

The *axiom of choice* is equivalent to the statement that all epimorphisms $f : A \to B$ in the category Set are *split*; i.e., there exists a map $g : B \to A$ such that $g;f = id_B$. As a consequence, each homomorphism between Σ-algebras maps closed subsets to closed subsets. In particular, we have

Lemma 3.2 (Closed Images)**.** *For any Σ-homomorphism $f \colon \mathcal{A} \to \mathcal{B}$, the (set-theoretic) image $f(A) \subseteq B$ of the carrier A of $\mathcal{A}$ is closed in $\mathcal{B}$. We denote by $f(\mathcal{A})$ the unique Σ-subalgebra of $\mathcal{B}$ with the carrier $f(A)$ and call it the* (set-theoretic) image of $\mathcal{A}$ with respect to the Σ-homomorphism f.

Lemma 3.2 is the last brick we need to conclude that the epic Σ-homomorphisms are exactly the surjective ones.

Corollary 3.7 (Epic $\cong$ Surjective)**.** *For any Σ-homomorphism $f \colon \mathcal{A} \to \mathcal{B}$ we have $f(\mathcal{A}) = \mathcal{R}(f, \mathcal{B})$; thus, $f \colon \mathcal{A} \to \mathcal{B}$ is an epimorphism in* $\mathsf{Alg}(\Sigma)$ *if, and only if, the map $f \colon A \to B$ is surjective.*

Proof. By Corollary 3.4 we have $f(A) \subseteq R(f, \mathcal{B})$. Lemma 3.2 gives us the Σ-subalgebra $f(\mathcal{A})$ of $\mathcal{B}$ at hand and ensures $f(A) \supseteq R(f, \mathcal{B})$ due to the construction of $R(f, \mathcal{B})$ in Definition 3.6. This gives us $f(\mathcal{A}) = \mathcal{R}(f, \mathcal{B})$ since two Σ-subalgebras of a Σ-algebra $\mathcal{B}$ are equal if, and only if, they do have the same carrier. By Proposition 3.4 we obtain $f \colon \mathcal{A} \to \mathcal{B}$ epic if, and only if, $\mathcal{B} = f(\mathcal{A}) = \mathcal{R}(f, \mathcal{B})$. $\mathcal{B} = f(\mathcal{A})$, however, means that f is surjective. □

Remark 3.5 (Stepwise Generation of Closed Subsets)**.** *There is another, more constructive, way to construct the closed sets $R(A, \mathcal{B})$. We start with A and add all the elements from B that we can reach by successively applying the operations in $\mathcal{B}$ to elements that have already been reached. A categorical analysis, formalization, and generalization of this stepwise iterative construction can be found in [12], for example.*

3.3. Terms and Term Algebras

We define terms as strings of symbols. To distinguish terms from metalevel expressions, such as $op^{\mathcal{A}}(a_1, \ldots, a_n)$, we will use angle bracket symbols "$\langle$","$\rangle$", instead of parentheses "(",")", to build terms. Moreover, we will use delimiter signs $\ulcorner \ldots \urcorner$ to indicate that the expression between the delimiters is a string. So the delimiter signs $\ulcorner \ldots \urcorner$ are not constituents of terms and we may just drop them if convenient. The following is a traditional inductive definition of terms similar to [13,14]:

Definition 3.8 (Terms)**.** *The set $T_\Sigma(X)$ of all* Σ-terms *over a set X of* variables *is the smallest set of strings of symbols such that*

Variables: $\ulcorner x \urcorner \in T_\Sigma(X)$*, for all* $x \in X$*;*

Constants: $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X)$*, for all* $\mathtt{c} \in OP$ *with* $I_{\mathtt{c}} = I_0$*;*

Operations: $\ulcorner \mathtt{op}\langle t_1, \ldots, t_n \rangle \urcorner \in T_\Sigma(X)$*, for all* $\mathtt{op} \in OP$ *with* $I_{\mathtt{op}} = I_n$*,* $n \geq 1$ *and all maps* $\boldsymbol{t} = (\ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner)$ *in* $T_\Sigma(X)^{I_n}$*.*

Note that the assignments $x \mapsto \ulcorner x \urcorner$, assigning to each variable the string consisting only of a single symbol denoting this variable, define an injective map $\eta_X \colon X \to T_\Sigma(X)$. Note, moreover, that in case $X = I_n = \{i_1, i_2, \ldots, i_n\}$, each operation symbol $\mathtt{op} \in OP$ is reborn as the Σ-term $\ulcorner \mathtt{op}\langle i_1, \ldots, i_n \rangle \urcorner \in T_\Sigma(I_n)$.

A term can be seen as a "tree-like computation scheme" and if we assign to variables certain values in an algebra, we can compute a value in this algebra following this computation scheme. Terms are constructed inductively, and thus we can also define this kind of evaluation of terms inductively.

Definition 3.9 (Evaluation of terms)**.** *For any set X of variables, any Σ-algebra $\mathcal{A} = (A, OP^{\mathcal{A}})$ and any map $\alpha \colon X \to A$ (called a* variable assignment*), we can inductively define a map* $\alpha^* \colon T_\Sigma(X) \to A$*:*

Variables: $\alpha^*(x) := \alpha(x)$*, for all* $\ulcorner x \urcorner \in T_\Sigma(X)$*;*

Constants: $\alpha^*(\mathtt{c}\langle\rangle) := \mathtt{c}^{\mathcal{A}}()(o)$*, for all* $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X)$*;*

Operations: $\alpha^*(\mathtt{op}\langle t_1, \ldots, t_n \rangle) := \mathtt{op}^{\mathcal{A}}(\alpha^*(t_1), \ldots, \alpha^*(t_n))(o)$*, for all* $\ulcorner \mathtt{op}\langle t_1, \ldots, t_n \rangle \urcorner \in T_\Sigma(X)$*.*

All three cases in Definition 3.9 are disjoint and terms are only equal if, and only if, they are equal as strings and thus α^* is uniquely defined.

There is no indication in Definitions 3.8 and 3.9 where the sets $T_\Sigma(X)$ of Σ-terms live and where the evaluation of Σ-terms takes place. A very common and powerful practice in Universal Algebra is to internalize Σ-terms as elements of carriers of Σ-algebras and to encode term evaluation through Σ-homomorphisms. First, we observe that the stepwise construction of terms in Definition 3.8 can be reflected by defining for each operation symbol in OP a corresponding (constructor) operation on $T_\Sigma(X)$:

Definition 3.10 (Term algebra)**.** *For a set X of variables, we define the term* Σ-algebra over X $\mathcal{T}_\Sigma(X) = (T_\Sigma(X), OP^{\mathcal{T}_\Sigma(X)})$ *by*

Constants: $\mathtt{c}^{\mathcal{T}_\Sigma(X)}()(o) := \ulcorner \mathtt{c}\langle\rangle \urcorner$*, for all* $\mathtt{c} \in OP$ *with* $I_{\mathtt{c}} = I_0$*, and*

Operations: $\mathtt{op}^{\mathcal{T}_\Sigma(X)}(\boldsymbol{t})(o) := \ulcorner \mathtt{op}\langle t_1, \ldots, t_n \rangle \urcorner$*, for all* $\mathtt{op} \in OP$ *with* $I_{\mathtt{op}} = I_n$*,* $n \geq 1$ *and all maps* $\boldsymbol{t} = (\ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner)$ *in* $T_\Sigma(X)^{I_n}$*.*

Note that the term Σ-algebra $\mathcal{T}_\Sigma(X)$ is generated by $\eta_X(X) \subseteq T_\Sigma(X)$. This is implicitly ensured by the statement of $T_\Sigma(X)$ being the *smallest* set satisfying the conditions in Definition 3.8. We say that the elements in $\eta_X(X)$ are the *generators* of $\mathcal{T}_\Sigma(X)$.

Second, we observe that the introduction of term Σ-algebras $\mathcal{T}_\Sigma(X)$ allows us to encode the defining equations for the cases **Constants** and **Operations** in Definition 3.9

via the requirement that the map $\alpha^*\colon\ T_\Sigma(X) \to A$ should establish a Σ-homomorphism $\alpha^*\colon\ \mathcal{T}_\Sigma(X) \to \mathcal{A}$:

Constants: $\alpha^*(\mathsf{c}\langle\rangle) = \alpha^*(\mathsf{c}^{\mathcal{T}_\Sigma(X)}()(o)) = (\mathsf{c}^{\mathcal{T}_\Sigma(X)}();\alpha^*)(o) = \mathsf{c}^{\mathcal{A}}()(o)$, for all $\ulcorner \mathsf{c}\langle\rangle \urcorner \in T_\Sigma(X)$;

Operations: $\alpha^*(\mathsf{op}\langle t_1,\ldots,t_n\rangle) = \alpha^*(\mathsf{op}^{\mathcal{T}_\Sigma(X)}(\boldsymbol{t})(o)) = (\mathsf{op}^{\mathcal{T}_\Sigma(X)}(\boldsymbol{t});\alpha^*)(o) = \mathsf{op}^{\mathcal{A}}(\boldsymbol{t};\alpha^*)(o)$ $= \mathsf{op}^{\mathcal{A}}(\alpha^*(t_1),\ldots,\alpha^*(t_n))(o)$, for all $\ulcorner \mathsf{op}\langle t_1,\ldots,t_n\rangle \urcorner \in T_\Sigma(X)$ where $\boldsymbol{t}$ is the map in $T_\Sigma(X)^{I_n}$ defined by $\boldsymbol{t} = (\ulcorner t_1 \urcorner,\ldots,\ulcorner t_n \urcorner)$.

The third case **Variables** in Definition 3.9 simply requires that the map $\alpha^*\colon\ T_\Sigma(X) \to A$ is an extension of the map $\alpha\colon\ X \to A$ and thus the statement "Definition 3.9 defines $\alpha^*\colon\ T_\Sigma(X) \to A$ uniquely" is transformed into the statement that the term Σ-algebra $\mathcal{T}_\Sigma(X)$ is a Σ-algebra *freely generated by* X.

Proposition 3.5 (Term Algebras as Free Construction)**.** *Given a set X of variables, the Σ-term algebra $\mathcal{T}_\Sigma(X)$ has the following universal property: For any Σ-algebra $\mathcal{A} = (A, OP^{\mathcal{A}})$ and any map $\alpha\colon\ X \to A$, there exists a unique Σ-homomorphism $\alpha^*\colon\ \mathcal{T}_\Sigma(X) \to \mathcal{A}$ such that $\eta_X;\alpha^* = \alpha$.*

$$\begin{tikzcd}
\mathsf{Set} & X \arrow[r, "\eta_X"] \arrow[dr, "\alpha"'] \arrow[r, phantom, shift right=6, "\circlearrowleft"] & T_\Sigma(X) \arrow[d, "\alpha^*"] & \mathcal{T}_\Sigma(X) \arrow[d, "\alpha^*"] & \mathsf{Alg}(\Sigma) \\
 & & A & \mathcal{A} &
\end{tikzcd}$$

The universal property in Proposition 3.5 characterizes $\mathcal{T}_\Sigma(X)$ uniquely up to isomorphism and the case $X = \emptyset$ gives us an initial Σ-algebra at hand.

Corollary 3.8. *$\mathcal{T}_\Sigma(\emptyset)$ is initial in the category* $\mathsf{Alg}(\Sigma)$*.*

It is a standard result for free constructions that the assignments $X \mapsto \mathcal{T}_\Sigma(X)$ and $(f : X \to Y) \mapsto ((f;\eta_Y)^* : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y))$ define a (free) functor $\mathbf{F}_\Sigma\colon\ \mathsf{Set} \to \mathsf{Alg}(\Sigma)$ that is *left-adjoint* to the forgetful functor $\mathbf{U}_\Sigma$ (see [15]).

$$\begin{tikzcd}
\mathsf{Set} \arrow[r, shift left, "\mathbf{F}_\Sigma"] & \mathsf{Alg}(\Sigma) \arrow[l, shift left, "\mathbf{U}_\Sigma"]
\end{tikzcd}$$

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "f"'] \arrow[dr, phantom, "\circlearrowleft"] & T_\Sigma(X) \arrow[d, "(f;\eta_Y)^*"] & \mathcal{T}_\Sigma(X) \arrow[d, "(f;\eta_Y)^*"] \\
Y \arrow[r, "\eta_Y"'] & T_\Sigma(Y) & \mathcal{T}_\Sigma(Y)
\end{tikzcd}$$

3.4. Substitutions

The very appealing advantage of internalizing terms as elements of carriers of algebras, encoding term evaluations as homomorphisms and thus having the adjunction $\mathbf{F}_\Sigma \dashv \mathbf{U}_\Sigma$ at hand, is that we obtain a fully fledged, well-defined, and well-behaved *substitution calculus* for free relying on general results in Category Theory. We insert an informal exposition of what we mean by a substitution calculus and what the expected features of such a calculus could be. The advantages of the "internal view of terms" will be discussed afterwards.

3.4.1. Substitution Calculi

The concept *substitution* is a kind of conceptual descendant of the concept of *variable*. A variable in an expression is a "free location" where we can put in expressions of a certain kind. The basic constituent of a substitution calculus is its specific way to describe a

(1) *Substitution (declaration)*, i.e., an assignment of expressions to variables.

In Universal Algebra, we can formalize substitutions as maps $\sigma\colon\ X \to T_\Sigma(Y)$. For finite sets $X = \{x_1, \ldots, x_n\}$ we may simply declare a substitution by listing the corresponding assignments $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

The second constituent of a substitution calculus is the specific mechanism for

(2) *Substitution application*, i.e., the replacement of occurrences of variables in a given expression by the expressions assigned to the variables by a substitution.

A common practice in Universal Algebra [14] is to denote the resulting term in $T_\Sigma(Y)$ of applying a finite substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ to a term t in $T_\Sigma(X)$ by

$$t[x_1/t_1, \ldots, x_n/t_n]. \tag{10}$$

An obvious, but not always trivial, requirement for substitution application is

(3) *Preservation of well formedness*; i.e., replacing variables in a well-formed expression by well-formed expressions should result in a well-formed expression.

In the case of terms, "well-formedness" simply means that we consider only those strings of symbols as terms that can be generated inductively by the three rules in Definition 3.8.

Let us assume that we have three collections of expressions and two *linkable substitutions*. The first substitution replaces variables in expressions from the first collection with expressions from the second collection, and thus its application produces expressions in the second collection. Analogously, the second substitution replaces variables in expressions from the second collection by expressions from the third collection and its application results in expressions from the third collection. In this situation, we do have two possibilities to transform expressions from the first collection into expressions from the third collection. First, we can apply both substitutions successively. Second, we can compose both "small step" substitutions into a single "big step" substitution. That is, we apply the second substitution to all the expressions appearing in the definition of the first substitution and obtain a new substitution replacing variables in expressions from the first collection by expressions from the third collection. This puts another feature of substitution calculi on the agenda:

(4) *Composition of linkable substitutions*.

The composition of the two linkable finite substitutions $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ from $X = \{x_1, \ldots, x_n\}$ to $T_\Sigma(Y)$ and $\{y_1 \mapsto r_1, \ldots, y_m \mapsto r_m\}$ from $Y = \{y_1, \ldots, y_m\}$ to $T_\Sigma(Z)$ results, for example, in the finite substitution

$$\{x_1 \mapsto t_1[y_1/r_1, \ldots, y_m/r_m], \ldots, x_n \mapsto t_n[y_1/r_1, \ldots, y_m/r_m]\} \tag{11}$$

from X to $T_\Sigma(Z)$.

Obviously, we would like that the application of the "big step" substitution always produces the same result as the successive application of the two linkable "small step" substitutions; i.e., for a substitution calculus, we require that

(5) *The composition of substitutions is compatible with substitution application.*

For the two linkable finite substitutions above, this requirement can be expressed by the equation

$$\begin{aligned} &t[x_1/t_1[y_1/r_1, \ldots, y_m/r_m], \ldots, x_n/t_n[y_1/r_1, \ldots, y_m/r_m]] \\ =&(t[x_1/t_1, \ldots, x_n/t_n])[y_1/r_1, \ldots, y_m/r_m] \end{aligned} \tag{12}$$

The compatibility of the composition of substitutions with substitution application usually ensures another useful property:

(6) *The composition of substitutions is associative.*

These are the six *syntactic features* we claim to be the essential characteristics of a substitution calculus as such. If a substitution calculus is, however, part of a bigger logic formalism where semantic structures are also considered, we will have some additional features concerning the interplay of syntax and semantics.

In analogy to substitutions, we first have to choose a way to describe

(7) *Variable assignments*, i.e., assignments of semantic items to variables.

In Universal Algebra, we work exclusively with variables ranging over elements in sets, and thus variable assignments can be defined as maps $\alpha\colon\ X \to A$ from a set of variables into the carrier set of a Σ-algebra $\mathcal{A}$, as in Definition 3.9.

In analogy to the step from substitution (declaration) to substitution application, each variable assignment should induce a corresponding

(8) *evaluation of expressions*, computing for each expression a unique semantic item or truth value.

Definition 3.9 presents, for example, an inductive definition of the evaluation $\alpha^*\colon\ T_\Sigma(X) \to A$ of Σ-terms into elements in the carrier A of a Σ-algebra $\mathcal{A}$ induced by a variable assignment $\alpha\colon\ X \to A$.

Since variable assignments establish a bridge from syntax to semantics, there is no composition of variable assignments in a substitution calculus. We should, however, have

(9) *The composition of substitutions with variable assignments* as well as *The composition of variable assignments with homomorphisms*.

For both new kinds of composition, it is desirable to have

(10) *Compatibility* with respect to substitution application and/or evaluation.

Finally, it would be reasonable to require

(11) *Associativity* for the three new possible combinations of the four kinds of composition, i.e., substitution–substitution–assignment, substitution–assignment–homomorphism, and assignment–homomorphism–homomorphism.

3.4.2. Substitutions by Algebraic Extensions

We now discuss the specialties of the "internalization approach" in view of the informal concept of a substitution calculus outlined in the last subsection.

Features (7) and (8): *Variable assignments* are formalized as maps $\alpha\colon\ X \to A$ from a set of variables into the carrier set A of a Σ-algebra $\mathcal{A}$ and the corresponding unique *term evaluations* are inductively defined maps $\alpha^*\colon\ T_\Sigma(X) \to A$ according to Definition 3.9.

Introducing term Σ-algebras and realizing that the inductive definition of unique term evaluations can be described as unique *algebraic extensions* $\alpha^*\colon\ \mathcal{T}_\Sigma(X) \to \mathcal{A}$ of variable assignments $\alpha\colon\ X \to A$, as stated in Proposition 3.5, has three immediate consequences.

Feature (1): *Substitutions* $\sigma\colon\ X \to T_\Sigma(Y)$ simply become a special case of variable assignments and

Feature (2): *Substitution applications* appear as a special case of term evaluations, namely as algebraic extensions $\sigma^*\colon\ \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$. Applying a substitution $\sigma\colon\ X \to T_\Sigma(Y)$ to a Σ-term $t \in T_\Sigma(X)$ means nothing but to compute the Σ-term $\sigma^*(t) \in T_\Sigma(Y)$. As mentioned before, it is also common to use instead of $\sigma^*(t)$ the more informative notation in (10) in the case of finite sets of variables.

Feature (3): *Preservation of well formedness* is implicitly ensured by the fact that the structures we define in Definition 3.10 (Term algebra) are indeed Σ-algebras.

Feature (9): The composition of a substitution $\sigma\colon\ X \to T_\Sigma(Y)$ with a variable assignment $\alpha\colon\ Y \to A$ is the variable assignment $\sigma;\alpha^*\colon\ X \to A$, while Feature (10), i.e., the compatibility of substitution application and evaluation, is ensured by the uniqueness of algebraic extensions: $\sigma^*;\alpha^* = (\sigma;\alpha^*)^*$ (see the left diagram below).

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\sigma"'] & T_\Sigma(X) \arrow[d, "\sigma^*"] \arrow[dl, phantom, "\circlearrowleft", pos=0.3] \arrow[dd, bend left=60, "(\sigma;\alpha^*)^*"] & \\
Y \arrow[r, "\eta_Y"] \arrow[dr, "\alpha"'] & T_\Sigma(Y) \arrow[d, "\alpha^*"] \arrow[dl, phantom, "\circlearrowleft", pos=0.3] & \circlearrowleft \\
 & A &
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "\sigma"'] & T_\Sigma(X) \arrow[d, "\sigma^*"] \arrow[dl, phantom, "\circlearrowleft", pos=0.3] \arrow[dd, bend left=60, "(\sigma;\delta^*)^*"] & \\
Y \arrow[r, "\eta_Y"] \arrow[dr, "\delta"'] & T_\Sigma(Y) \arrow[d, "\delta^*"] \arrow[dl, phantom, "\circlearrowleft", pos=0.3] & \circlearrowleft \\
 & T_\Sigma(Z) &
\end{tikzcd}$$

Feature (4), *composition of linkable substitutions*, becomes a special case of Feature (9): The composition of a substitution $\sigma\colon X \to T_\Sigma(Y)$ and a substitution $\delta\colon Y \to T_\Sigma(Z)$ is the substitution $\sigma;\delta^*\colon X \to T_\Sigma(Z)$. In this way, Feature (5), *composition of substitutions is compatible with substitution application*, becomes a special case of Feature (10): $\sigma^*;\delta^* = (\sigma;\delta^*)^*$ (see the right diagram above), and 12 spells out this equation for the finite case.

The remaining part of Features (9) and (10) is also ensured by Proposition 3.5: The composition of a variable assignment $\alpha\colon X \to A$ with a Σ-homomorphism $h : \mathcal{A} \to \mathcal{B}$, for example, is the variable assignment $\alpha;h : X \to B$ and the uniqueness of algebraic extensions ensures compatibility: $\alpha^*;h = (\alpha;h)^*$.

Finally, all the compatibilities together with the associativity of composition of maps also gives us the three kinds of associativity required by Feature (11). The proof of the associativity

substitution ; (substitution ; assignment) = (substitution ; substitution) ; assignment,

for example, is simply given by the compatibility of substitution application and evaluation, as well as associativity of map composition: $\sigma;(\delta;\beta^*)^* = \sigma;(\delta^*;\beta^*) = (\sigma;\delta^*);\beta^*$ for arbitrary substitutions $\sigma\colon X \to T_\Sigma(Y)$, $\delta\colon Y \to T_\Sigma(Z)$ and assignments $\beta\colon Z \to A$.

Remark 3.6 (No Internalization)**.** *Of course, there is no need to utilize internalization of terms and uniqueness of algebraic extensions to establish a fully fledged substitution calculus for Universal Algebra! Instead, we could just separately work out each of the necessary definitions and proofs based on the inductive definition of terms analogous to Definition 3.9 (Evaluation of Terms). Internalization simply saves us a lot of work if it comes to substitutions!*

On the other hand, internalization is obviously not helpful in establishing substitution calculi for pure syntactic logic frameworks and/or for logic frameworks without operations. Also in logic frameworks, where some variables may range over logic formulas, the internalization of terms will be of restricted help.

As an example of a definition that is independent of Proposition 3.5, we give an explicit inductive definition of substitution application.

Definition 3.11 (Substitution Application)**.** *For any sets X, Y of variables and any substitution $\delta\colon X \to T_\Sigma(Y)$, we can inductively define a corresponding* substitution application *$\delta^*\colon T_\Sigma(X) \to T_\Sigma(Y)$ such that $\eta_X;\delta^* = \delta$:*

Variables: $\delta^*(x) := \delta(x)$, *for all* $\ulcorner x \urcorner \in T_\Sigma(X)$;

Constants: $\delta^*(\mathtt{c}\langle\rangle) := \mathtt{c}\langle\rangle$, *for all* $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X)$;

Operations: $\delta^*(\mathtt{op}\langle t_1,\ldots,t_n\rangle) := \mathtt{op}\langle \delta^*(t_1),\ldots,\delta^*(t_n)\rangle$, *for all* $\ulcorner \mathtt{op}\langle t_1,\ldots,t_n\rangle \urcorner \in T_\Sigma(X)$.

Remark 3.7 (Kleisli Category)**.** *The composition of substitutions defined above, together with η_X as the identity substitution for every set X, gives us a category of substitutions. In more abstract categorical terms, this is exactly the* Kleisli category *of the adjunction $\mathbf{F}_\Sigma \dashv \mathbf{U}_\Sigma$, which is equivalent to the full subcategory of $\mathsf{Alg}(\Sigma)$ of all term Σ-algebras (see, for example, [15]).*

3.5. Two Model-Theoretic Implications of the Existence of a Free Functor

Before we turn to our actual topic of "derived operations", it is maybe worth rounding up our discussion of monomorphisms and epimorphisms in $\mathsf{Alg}(\Sigma)$. We have already mentioned in Corollary 3.2 that injective Σ-homomorphisms are monomorphisms in $\mathsf{Alg}(\Sigma)$ since the functor $\mathbf{U}_\Sigma : \mathsf{Alg}(\Sigma) \to \mathsf{Set}$ is faithful and therefore reflects monomorphisms. The first observation is that the existence of the free functor $\mathbf{F}_\Sigma\colon \mathsf{Set} \to \mathsf{Alg}(\Sigma)$ now provides the implication in the other direction, since right-adjoint functors preserve monomorphisms.

Lemma 3.3 (Monic implies Injective)**.** *For every monic Σ-homomorphism $f\colon \mathcal{A} \to \mathcal{B}$, the underlying map $f : A \to B$ is a monomorphism in* Set*, i.e., injective.*

Proof. We consider an arbitrary set X and arbitrary maps $g, h : X \to A$ such that $g; f = h; f$.

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, shift left, "h"] \arrow[dr, shift right, "g"'] \arrow[ddr, bend right, "g;f=h;f"'] & T_\Sigma(X) \arrow[d, shift right, "g^*"'] \arrow[d, shift left, "h^*"] \arrow[dd, bend left=70, "(g;f)^*=(h;f)^*"] \arrow[dl, phantom, "\circlearrowleft"] \\
& A \arrow[d, "f"] \arrow[r, phantom, "\circlearrowleft"] & {} \\
& B
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{T}_\Sigma(X) \arrow[d, shift right, "g^*"'] \arrow[d, shift left, "h^*"] \arrow[dd, bend left=70, "(g;f)^*=(h;f)^*"] \\
\mathcal{A} \arrow[d, "f"] \arrow[r, phantom, "\circlearrowleft"] & {} \\
\mathcal{B}
\end{tikzcd}$$

By the uniqueness of algebraic extensions, we have $g^*; f = (g; f)^*$ and $h^*; f = (h; f)^*$ (Feature (10), compatibility of substitution application and evaluation), and thus the assumption entails $g^*; f = (g; f)^* = (h; f)^* = h^*; f$. This implies $g^* = h^*$ since $f\colon \mathcal{A} \to \mathcal{B}$ is monic. By pre-composition with η_X, we obtain $g = \eta_X; g^* = \eta_X; h^* = h$ as required. □

Second, the free functor helps to elucidate the intuition behind the choice of the adjectives "accessible"/"reachable" in Definition 3.6, namely that each element in $R(f, \mathcal{B})$ can be accessed/reached by first applying the map f and then successively applying the operations in $\mathcal{B}$.

Lemma 3.4 (Accessible via Map $\cong$ Accessible via Extension)**.** *For any Σ-algebra $\mathcal{B}$ and any map $f\colon A \to B$ we have $\mathcal{R}(f, \mathcal{B}) = \mathcal{R}(f^*, \mathcal{B})$ for the algebraic extension $f^*\colon T_\Sigma(A) \to B$. In such a way, $\mathcal{B}$ is accessible via $f\colon A \to B$ if, and only if, $\mathcal{B}$ is accessible via $f^*\colon T_\Sigma(A) \to B$.*

Proof. Feature (10), *compatibility of substitution application and evaluation*, ensures for any Σ-subalgebras $\mathcal{X}$ of $\mathcal{B}$ that f factors through $\iota_{X,B} : X \hookrightarrow B$ if, and only if, f^* factors through $\iota_{X,B} : X \hookrightarrow B$. □

By Lemma 3.4, Proposition 3.4, and Corollary 3.7, we obtain the following equivalences.

Corollary 3.9. *For any Σ-algebra $\mathcal{B}$ and any map $f : A \to B$, the following statements are equivalent:*

1. *$\mathcal{B}$ is accessible via $f : A \to B$.*
2. *$\mathcal{B}$ is accessible via $f^* : T_\Sigma(A) \to B$.*
3. *$f^* : \mathcal{T}_\Sigma(A) \to \mathcal{B}$ is an epimorphism in* $\mathsf{Alg}(\Sigma)$.
4. *$f^* : T_\Sigma(A) \to B$ is an epimorphism in* Set*, i.e., surjective.*

3.6. Terms and Derived Operations

For any set X and any Σ-algebra $\mathcal{A}$ the evaluation of Σ-terms over X in $\mathcal{A}$ is actually a map from $T_\Sigma(X) \times A^X$ into A. In Definition 3.9, we fix an arbitrary element $\alpha \in A^X$ and define a corresponding map $\alpha^*\colon T_\Sigma(X) \to A$ by varying inductively over $T_\Sigma(X)$. That is, we describe the map from $T_\Sigma(X) \times A^X$ into A through an A^X-indexed family of maps $\alpha^* \colon T_\Sigma(X) \to A$. This kind of splitting is the basis for the internalization trick.

We can, however, also proceed the other way around. We can represent the map from $T_\Sigma(X) \times A^X$ into A via a $T_\Sigma(X)$-indexed family of maps from A^X into A or A^O. $O = \{o\}$ is the singleton used in Definition 3.1 to declare the output arity of operation symbols.

Definition 3.12 (Derived Operations)**.** *For any set X of variables, any Σ-algebra $\mathcal{A}$ and any Σ-term $t \in T_\Sigma(X)$, we define a corresponding* derived operation, *i.e., the map*

$$t^{\mathcal{A}} \colon A^X \to A^O \quad \textit{with } t^{\mathcal{A}}(\alpha)(o) := \alpha^*(t) \textit{ for all } \alpha \in A^X.$$

We call the maps $t^{\mathcal{A}}$ *derived operations* since they are built up from the *basic operations* in $OP^{\mathcal{A}}$ (compare Definition 3.14 below). Derived operations live on the same "external level" as the basic operations, i.e., outside of carrier sets of algebras. Terms represent those derived operations, and thus, it is opportune to also have a complementary *external view* on terms and consider them as syntactic entities living together with operation symbols on the same external level. In particular, we can consider terms as entities existing independent of and prior to algebras.

To support and validate the external view on terms, we should avoid the sleight of hand in Definition 3.12 and define derived operations, independent of Definition 3.9, simply by constructing new maps from given maps.

The only two constructions we need for this purpose are available in any category with finite products: the composition of maps (morphisms) and the tupling of maps (morphisms). Since we use non-traditional finite products A^I, instead of traditional Cartesian products A^n, to define domains and codomains of operations, it is probably worth it to spell out the corresponding version of tupling we will rely on.

Definition 3.13 (Tupling of Maps)**.** *For any family of maps $f_j \colon A^X \to A^O$ with $1 \le n, 1 \le j \le n$ we can construct a map $\langle\!\langle f_1, \ldots, f_n \rangle\!\rangle \colon A^X \to A^{I_n}$, $I_n := \{i_1, i_2, \ldots, i_n\}$ defined by*

$$\langle\!\langle f_1, \ldots, f_n \rangle\!\rangle(\alpha)(i_j) := f_j(\alpha)(o) \quad \textit{for all } \alpha \in A^X \textit{ and all } 1 \le j \le n.$$

For an empty family of maps, $\langle\!\langle\, \rangle\!\rangle \colon A^X \to A^{I_0}$ denotes the constant map assigning to all $\alpha \in A^X$ the only element in A^{I_0} represented by the empty tuple $()$.

Now, we are prepared to give an inductive definition of derived operations. The base cases are projection maps, represented by variables and constant maps. The induction step is implicitly divided into two steps: first tupling and then composition with a basic operation.

Definition 3.14 (Construction of Derived Operations)**.** *For any set X and any Σ-algebra $\mathcal{A}$ we define inductively for all Σ-terms $t \in T_\Sigma(X)$ a corresponding* derived operation $t^{\mathcal{A}} \colon A^X \to A^O$ *as follows:*

Variables: *for all $\ulcorner x \urcorner \in T_\Sigma(X)$, the (projection) map $x^{\mathcal{A}} \colon A^X \to A^O$ is defined by $x^{\mathcal{A}}(\alpha)(o) := \pi_x(\alpha) = \alpha(x)$ for all $\alpha \in A^X$;*

Constants: $\mathtt{c}\langle\rangle^{\mathcal{A}} := \langle\!\langle\, \rangle\!\rangle ; \mathtt{c}^{\mathcal{A}}$, *for all $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X)$;*

Operations: $\mathtt{op}\langle t_1, \ldots, t_n \rangle^{\mathcal{A}} := \langle\!\langle t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}} \rangle\!\rangle ; \mathtt{op}^{\mathcal{A}}$, *for all $\ulcorner \mathtt{op}\langle t_1, \ldots, t_n \rangle \urcorner \in T_\Sigma(X)$, $n \ge 1$.*

3.7. Syntactic Lawvere Theories

As long as we restrict ourselves to finite sets of variables, syntactic Lawvere theories are the ultimate implementation of the external view on terms while also incorporating the internal view, as we will soon see. We will not give a fully detailed exposition but just enough to be prepared for the discussion and definition of derived graph operations in Section 5 (the interested reader may consult [8] for more details).

3.7.1. Construction of Lawvere Theories

Relying on the concept of Σ-term and a substitution calculus, as discussed in the last section, we can define for any signature $\Sigma = (OP, ar)$ a syntactic category $\mathsf{L}(\Sigma)$ as follows:

Objects: As objects, we choose canonical finite sets of variables

$$X_0 = \varnothing \quad \text{and} \quad X_n := \{x_1^n, x_2^n, \ldots, x_n^n\} \text{ for all } n \in \mathbb{N} \text{ with } n \geq 1\,. \tag{13}$$

For all $n \geq 2$, we assume X_n to be equipped with a fixed total order $x_1^n < x_2^n < \ldots < x_n^n$; thus, we can reuse the tuple notation to represent maps as discussed in Section 2.

Morphisms: Morphisms are all tuples $(t_1, \ldots, t_n) : X_m \to X_n$ representing a substitution (declaration) $\boldsymbol{t} : X_n \to T_\Sigma(X_m)$.

Identities: The identity on X_n is the tuple $(x_1^n, \ldots, x_n^n) : X_n \to X_n$ representing the substitution $\eta_{X_n} : X_n \to T_\Sigma(X_n)$.

Composition: The composition of two tuples $(r_1, \ldots, r_m) : X_k \to X_m$ and $(t_1, \ldots, t_n) : X_m \to X_n$ is the tuple $(\boldsymbol{r}^*(t_1), \ldots, \boldsymbol{r}^*(t_n)) : X_k \to X_n$ representing the substitution $\boldsymbol{t}; \boldsymbol{r}^* : X_n \to T_\Sigma(X_k)$ where $\boldsymbol{r}^* : T_\Sigma(X_m) \to T_\Sigma(X_k)$ is the application of the substitution $\boldsymbol{r} : X_m \to T_\Sigma(X_k)$ according to Definition 3.11 (compare also (11)).

Laws: Identity and associativity law are ensured by Feature (5), *composition of substitutions is compatible with substitution application* (compare also (12)).

3.7.2. Properties of Lawvere Theories

The category $\mathsf{L}(\Sigma)$ has all finite products. We describe binary products as follows:

- The product of two objects X_n and X_m is defined by $X_n \times X_m := X_{n+m}$ with projections $\pi_1 := (x_1^{n+m}, \ldots, x_n^{n+m}) : X_{n+m} \to X_n$ and $\pi_2 := (x_{n+1}^{n+m}, \ldots, x_{n+m}^{n+m}) : X_{n+m} \to X_m$.
- The tuple of two morphisms $(t_1, \ldots, t_n) : X_k \to X_n$ and $(r_1, \ldots, r_m) : X_k \to X_m$ in $\mathsf{L}(\Sigma)$ is given by

$$\langle\!\langle (t_1, \ldots, t_n), (r_1, \ldots, r_m) \rangle\!\rangle := (t_1, \ldots, t_n, r_1, \ldots, r_m) : X_k \to X_{n+m}\,.$$

Remark 3.8 (Product versus Sum)**.** *The tentative reader has surely realized that X_{n+m} is not the product but the sum of X_n and X_m in the category* Set *and that $(t_1, \ldots, t_n, r_1, \ldots, r_m)$ represents the cotuple $[\boldsymbol{t}, \boldsymbol{r}] : X_{n+m} \to T_\Sigma(X_k)$ of the two maps $\boldsymbol{t} : X_n \to T_\Sigma(X_k)$ and $\boldsymbol{r} : X_m \to T_\Sigma(X_k)$ in* Set*. However, by choosing the direction of morphisms $(t_1, \ldots, t_n) : X_k \to X_n$ in $\mathsf{L}(\Sigma)$ in accordance with the direction of their semantics, $\langle\!\langle t_1^A, \ldots, t_n^A \rangle\!\rangle : A^{X_k} \to A^{X_n}$, X_{n+m} indeed becomes the categorical product of X_n and X_m in $\mathsf{L}(\Sigma)$. In other words, it is much more convenient for us to describe $\mathsf{L}(\Sigma)$ as a category with finite products instead of a category with finite sums. In this way, we avoid, especially, the needless use of opposite categories.*

Nevertheless, the reader should keep in mind that syntactic entities are usually and most conveniently constructed by colimits in Set *while the* semantics as interpretation paradigm *turns those colimits on the syntactic level into corresponding limits on the semantic level. We do have, for example, the exponential law $A^{X_{n+m}} \cong A^{X_n} \times A^{X_m}$ with "$\_ \times \_$" denoting this time the Cartesian product of sets.*

The construction of syntactic Lawvere theories $\mathsf{L}(\Sigma)$ for signatures Σ is a free construction on the "external meta-level". More specifically, there is, for any signature Σ an equivalence between the category $\mathsf{Alg}(\Sigma)$ and the category of all finite product-preserving functors from $\mathsf{L}(\Sigma)$ into Set: for every finite product preserving functor $\mathbf{H} : \mathsf{L}(\Sigma) \to \mathsf{Set}$, there is a corresponding Σ-algebra $\mathcal{U}(\mathbf{H})$ with carrier $\mathbf{H}(X_1)$ and operations defined for each n-ary operation symbol in OP by $\mathtt{op}^{\mathcal{U}(\mathbf{H})} := b_n; \mathbf{H}((\mathtt{op}\langle x_1^n, \ldots, x_n^n \rangle)); b$ with the isomorphisms $b_n : \mathbf{H}(X_1)^{I_n} \to \mathbf{H}(X_n)$, provided by the assumption that $\mathbf{H}$ preserves finite products, and the obvious isomorphism $b : \mathbf{H}(X_1) \to \mathbf{H}(X_1)^O$.

Conversely, for every Σ-algebra $\mathcal{A}$ the assignments $X_1 \mapsto A$, $X_n \mapsto A^{I_n}$ for all $n \geq 2$, and $(t_1, \ldots, t_n) \mapsto \langle\!\langle t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}} \rangle\!\rangle$ give rise to a unique finite limit preserving functor $\mathbf{FP}(\mathcal{A}) : \mathsf{L}(\Sigma) \to \mathsf{Set}$ such that $\mathcal{U}(\mathbf{FP}(\mathcal{A})) = \mathcal{A}$. This is ensured by Definitions 3.13 and 3.14.

Finally, $\mathsf{L}(\Sigma)$ comprises all finite term Σ-algebras in the following sense: it is well-known that hom-functors preserve limits and thus, especially, finite products. In such a way, for all $n \in \mathbb{N}$ the hom-functor $\mathsf{L}(\Sigma)(X_n, \_) : \mathsf{L}(\Sigma) \to \mathsf{Set}$ preserves finite limits. According to the above equivalence of categories and the *Yoneda Lemma*, the corresponding Σ-algebra $\mathcal{U}(\mathsf{L}(\Sigma)(X_n, \_))$ satisfies the universal property stated in Proposition 3.5. This means, however, nothing but the Σ-algebras $\mathcal{U}(\mathsf{L}(\Sigma)(X_n, \_))$ and $\mathcal{T}_\Sigma(X_n)$ being isomorphic.

4. Graph Algebras and Graph Term Algebras

Relying on the categorical reconstruction of concepts, the constructions and results of traditional Universal Algebra in Section 3, we present in this section a generalization of those concepts, constructions and results to graph algebras.

4.1. Graph Signatures, Graph Algebras, and Homomorphisms

We consider the composition of two morphisms in a category as a graph operation. The arity of a corresponding operation symbol `comp` could be declared in the following way.

$$\mathtt{I}_{\mathtt{comp}} : \; iv_1 \xrightarrow{ie_1} iv_2 \xrightarrow{ie_2} iv_3 \qquad\qquad \mathtt{O}_{\mathtt{comp}} : \; iv_1 \xrightarrow{oe_1} iv_3 \tag{14}$$

Compared to traditional algebraic operations, we can presently infer some essential differences [7]:

Two different kinds of input items. This is evident due to working with graphs that consist of both vertices and edges. Instead of a single set, we therefore declare a graph as the input arity.

Arbitrarily many output items. A single output is assumed for algebraic operations, but graph operations can produce arbitrarily large finite graphs as output. Similar to the input, the output arity is chosen to be a graph.

Output is often related to the input. In the case of the composition operation `comp` above, the relation between the two arity graphs $\mathtt{I}_{\mathtt{comp}}$ and $\mathtt{O}_{\mathtt{comp}}$ is clear from the labelling: the output edge oe_1 has the same source and target as the input edges ie_1 and ie_2, respectively. Instead of always requiring coherent labeling, we introduce a third arity graph $\mathtt{B}_{\mathtt{comp}}$, called the *boundary* of `comp`, to encompass the connection between input and output in a fitting way.

We summarize the previous discussion as the following definition.

Definition 4.1 (Graph signature). *A graph signature $\Gamma = (OP, ar)$ is given by*

- *A set OP of operation symbols,*
- *A map ar assigning to each operation symbol* `op` *in OP its* arity span, *i.e., a span $ar(\mathtt{op}) =$ $\mathtt{I}_{\mathtt{op}} \xhookleftarrow{l_{\mathtt{op}}} \mathtt{B}_{\mathtt{op}} \xhookrightarrow{r_{\mathtt{op}}} \mathtt{O}_{\mathtt{op}}$ of inclusion graph homomorphisms between finite graphs such that the sets $(\mathtt{O}_{\mathtt{op}})_V \setminus (\mathtt{B}_{\mathtt{op}})_V$ and $(\mathtt{I}_{\mathtt{op}})_V$ as well as the sets $(\mathtt{O}_{\mathtt{op}})_E \setminus (\mathtt{B}_{\mathtt{op}})_E$ and $(\mathtt{I}_{\mathtt{op}})_E$ are disjoint. The graphs $\mathtt{I}_{\mathtt{op}}$, $\mathtt{B}_{\mathtt{op}}$, and $\mathtt{O}_{\mathtt{op}}$ are referred to as the* input arity, boundary arity, *and* output arity *of* `op`, *respectively.*

If $\mathtt{I}_{\mathtt{c}}$ is the empty graph $\mathbf{0}$ for $\mathtt{c} \in OP$, we also say that `c` *is a* constant symbol. *Note that $\mathtt{B}_{\mathtt{c}} = \mathbf{0}$ in this case.*

It is maybe worth mentioning that the disjointedness condition is equivalent to the condition that $\mathtt{B}_{\mathtt{op}}$ is the componentwise set intersection of the graphs $\mathtt{I}_{\mathtt{op}}$ and $\mathtt{O}_{\mathtt{op}}$, i.e.,

$$(\mathtt{B}_{\mathtt{op}})_V = (\mathtt{I}_{\mathtt{op}})_V \cap (\mathtt{O}_{\mathtt{op}})_V \quad \text{and} \quad (\mathtt{B}_{\mathtt{op}})_E = (\mathtt{I}_{\mathtt{op}})_E \cap (\mathtt{O}_{\mathtt{op}})_E. \tag{15}$$

Remark 4.1 (Arity Renamings). *In contrast to traditional operations, we use explicit names to identify the input and output "positions" of a graph operation. Names can, however, be chosen arbitrarily, and we should be prepared to rename, if necessary, the arities of a graph operation.*

An arity renaming *ϱ from an arity span $\mathtt{I} \xhookleftarrow{l} \mathtt{B} \xhookrightarrow{r} \mathtt{O}$ to another arity span $\mathtt{I}' \xhookleftarrow{l'} \mathtt{B}' \xhookrightarrow{r'} \mathtt{O}'$ is simply a triple of graph isomorphisms $\varrho_I : \mathtt{I} \to \mathtt{I}'$, $\varrho_B : \mathtt{B}_{\mathtt{op}} \to \mathtt{B}'$, $\varrho_O : \mathtt{O} \to \mathtt{O}'$ such that the following diagram commutes.*

$$\begin{tikzcd}
\mathtt{I} \arrow[d, "\varrho_I"'] \arrow[rd, phantom, "\circlearrowleft"] & \mathtt{B} \arrow[l, hook', "l"'] \arrow[r, hook, "r"] \arrow[d, "\varrho_B"'] \arrow[rd, phantom, "\circlearrowleft"] & \mathtt{O} \arrow[d, "\varrho_O"'] \\
\mathtt{I}' & \mathtt{B}' \arrow[l, hook', "l'"'] \arrow[r, hook, "r'"] & \mathtt{O}'
\end{tikzcd}$$

Remark 4.2 (Notational Conventions). *For all finite graphs $\mathtt{J}$ used in arity spans, we assume that the corresponding sets $\mathtt{J}_V$ and $\mathtt{J}_E$ are equipped with a fixed total order. Relying on our conventions in Section 2, this allows us to represent any graph homomorphism $\boldsymbol{b} = (\boldsymbol{b}_V, \boldsymbol{b}_E) : \mathtt{J} \to \mathtt{G}$ as a pair of tuples $\boldsymbol{b}_V = (bv_1, \ldots, bv_n)$, $\boldsymbol{b}_E = (be_1, \ldots, be_m)$, where $n = |\mathtt{J}_V|$, $m = |\mathtt{J}_E|$ and bv_i (be_j) the image of the i-th (j-th) element in $\mathtt{J}_V$ ($\mathtt{J}_E$), $1 \leq i \leq n$, $1 \leq j \leq m$. For any arity span $\mathtt{I} \xhookleftarrow{l} \mathtt{B} \xhookrightarrow{r} \mathtt{O}$, we assume that $\mathtt{B}$ inherits the order from $\mathtt{I}$ and $\mathtt{O}$, respectively.*

We impose the disjointness condition in Definition 4.1 to syntactically distinguish between input items of a graph operation and the new output items produced by a graph operation. Another objective is to be able to later infer the arity of a "graph operation expression" based only on the expression itself and the arities of the operations symbols defined in the corresponding signature.

To achieve this goal, we will use "canonical arity spans" to describe the arities of operation symbols and graph operation expressions. In a canonical arity span, *we use canonical sets $\mathtt{I}_V = \{iv_1, \ldots, iv_{n^{\mathtt{I}}}\}$ of input vertices and $\mathtt{I}_E = \{ie_1, \ldots, ie_{m^{\mathtt{I}}}\}$ of input edges with $n^{\mathtt{I}} = |\mathtt{I}_V|$ and $m^{\mathtt{I}} = |\mathtt{I}_E|$. In the same way, we use canonical sets $\mathtt{O}_V \setminus \mathtt{B}_V = \{ov_1, \ldots, ov_{n^{\mathtt{O}}}\}$ of output vertices and $\mathtt{O}_E \setminus \mathtt{B}_E = \{oe_1, \ldots, oe_{m^{\mathtt{O}}}\}$ of output edges with $n^{\mathtt{O}} = |\mathtt{O}_V \setminus \mathtt{B}_V|$ and $m^{\mathtt{O}} = |\mathtt{O}_E \setminus \mathtt{B}_E|$.*

Example 4.1 (Signature for Categories). *We define a signature Γ_{cat} with an operation symbol* `comp` *to denote operations composing two edges and an operation symbol* `id` *to denote operations, assigning to vertices corresponding identity edges.*

As per Definition 4.1, we extend the arity of `comp` *proposed in (14) to the span of graph homomorphisms shown in Figure 1 with a boundary graph $\mathtt{B}_{\mathtt{comp}}$ consisting of only two vertices, iv_1 and iv_3. This encapsulates exactly the desired requirements for a composition operation with regards to sources and targets.*

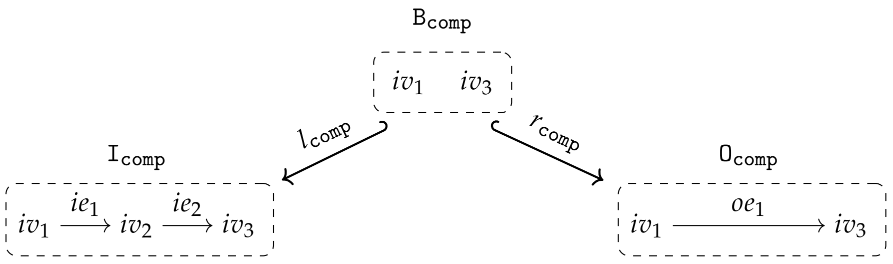


Figure 1. Arity declaration for the operation symbol `comp`.

Analogously, the arity of `id`*, shown in Figure 2, encapsulates the requirements for an identity operation with regard to sources and targets.*

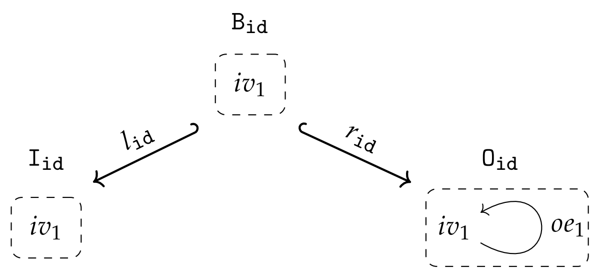


Figure 2. Arity declaration for the operation symbol id.

Graph is a locally small category, and we employ here the same exponential notation for hom-sets as we did for sets in Section 3. That is, for any graphs $\mathtt{G}$ and $\mathtt{H}$, $\mathtt{G}^{\mathtt{H}}$ denotes the set $\mathsf{Graph}(\mathtt{H},\mathtt{G})$ of all graph homomorphisms from $\mathtt{H}$ into $\mathtt{G}$.

Graph operations are maps, i.e., morphisms in Set! Specifically, an operation $\mathtt{op}^{\mathtt{G}}$ on a graph $\mathtt{G}$ should take as input a graph homomorphism in $\mathtt{G}^{\mathtt{I_{op}}}$ and return as output a graph homomorphism in $\mathtt{G}^{\mathtt{O_{op}}}$. This procedure needs to respect the boundary, which is ensured by requiring the resulting square being commutative.

Definition 4.2 (Graph Algebra)**.** *Let $\Gamma = (OP, ar)$ be a graph signature. A* (graph) Γ-algebra *$\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ is given*

- *By a graph $\mathtt{G}$, called the* carrier *of $\mathcal{G}$, and*
- *By a family $OP^{\mathcal{G}} = (\mathtt{op}^{\mathcal{G}} \colon \mathtt{G}^{\mathtt{I_{op}}} \to \mathtt{G}^{\mathtt{O_{op}}} \mid \mathtt{op} \in OP)$ of maps such that the following diagram commutes for all $\mathtt{op}$ in OP and all graph homomorphisms $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I_{op}}}$.*

$$\begin{tikzcd} & \mathtt{B_{op}} \arrow[dl, hook', "l_{op}"'] \arrow[dr, hook, "r_{op}"] & \\ \mathtt{I_{op}} \arrow[dr, "\boldsymbol{b}"'] & \circlearrowleft & \mathtt{O_{op}} \arrow[dl, "\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})"] \\ & \mathtt{G} & \end{tikzcd} \tag{16}$$

The maps in $OP^{\mathcal{G}}$ are referred to as graph operations.

In the specific case where $\mathtt{I_{op}}$ is the empty graph $\mathbf{0}$, the set $\mathtt{G}^{\mathtt{I_{op}}}$ becomes a singleton, as there is exactly one graph homomorphism $\mathbf{0}_{\mathtt{G}} : \mathbf{0} \to \mathtt{G}$, represented by a pair of empty tuples $\mathbf{0}_{\mathtt{G}} = ((),())$. Thus, for any constant symbol $\mathtt{c}$ in OP, with $\mathtt{I_c} = \mathbf{0}$, the corresponding graph operation $\mathtt{c}^{\mathcal{G}}$ returns a subgraph of $\mathtt{G}$, i.e., it returns the image of $\mathtt{O_c}$ under $\mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{G}})$.

In the case where the signature Γ has no constant symbols, the empty graph constitutes a Γ-algebra, called the *empty Γ-algebra*.

Example 4.2 (Categories as Graph Algebras)**.** *We consider the graph signature Γ_{cat} in Example 4.1.*

$$\begin{tikzcd} & \mathtt{B_{comp}} \arrow[dl, hook', "l_{comp}"'] \arrow[dr, hook, "r_{comp}"] & \\ \mathtt{I_{comp}} \arrow[dr, "\boldsymbol{b}"'] & \circlearrowleft & \mathtt{O_{comp}} \arrow[dl, "\mathtt{comp}^{\mathcal{C}}(\boldsymbol{b})"] \\ & gr(\mathsf{C}) & \end{tikzcd}$$

Obviously, any small category C gives rise to a Γ_{cat}-algebra $\mathcal{C} = (gr(\mathsf{C}), OP^{\mathcal{C}})$ with the underlying graph $gr(\mathsf{C})$ of C as carrier: the graph operation $\mathtt{comp}^{\mathcal{C}} \colon gr(\mathsf{C})^{\mathtt{I_{comp}}} \to gr(\mathsf{C})^{\mathtt{O_{comp}}}$ is defined *by the single equation* $(\texttt{comp}^{\mathcal{C}}(\boldsymbol{b}))_E(oe_1) := \boldsymbol{b}_E(ie_1);^{\mathsf{C}}\,\boldsymbol{b}_E(ie_2)$ *for all* $\boldsymbol{b} \in gr(\mathsf{C})^{\mathtt{I}_{\mathtt{comp}}}$ *where "*$\_;^{\mathsf{C}}\,\_$*" denotes the composition in* C*. For the two vertices in* $\mathtt{O}_{\mathtt{comp}}$*, the images with respect to* $\texttt{comp}^{\mathcal{C}}(\boldsymbol{b})$ *are always fixed due to the commutativity condition in Definition 4.2:* $(\texttt{comp}^{\mathcal{C}}(\boldsymbol{b}))_V(iv_1) = \boldsymbol{b}_V(iv_1)$ *and* $(\texttt{comp}^{\mathcal{C}}(\boldsymbol{b}))_V(iv_2) = \boldsymbol{b}_V(iv_2)$*.*

Not every Γ_{cat}*-algebra, however, can be seen as a category, since it may fail to satisfy the identity and/or the associativity law. In [7], we presented some ideas concerning equations for graph algebras, but the development of a full equational calculus is a topic for future research.*

Example 4.3 (Chosen Pullbacks)**.** *Graph algebras can serve as a conceptual tool to give a precise meaning to statements like "let* C *be a category with chosen pullbacks".*

We define the arity of an operation symbol $\texttt{pb}$ *as the span of inclusion graph homomorphisms shown in Figure 3. To choose pullbacks for a small category* C *then means nothing but to define a graph operation* $\texttt{pb}^{\mathcal{C}} : gr(\mathsf{C})^{\mathtt{I}_{\mathtt{pb}}} \to gr(\mathsf{C})^{\mathtt{O}_{\mathtt{pb}}}$ *assigning to each cospan* $\boldsymbol{b} : \mathtt{I}_{\mathtt{pb}} \to gr(\mathsf{C})$ *in* C *a corresponding pullback span* $\texttt{pb}^{\mathcal{C}} : \mathtt{O}_{\mathtt{pb}} \to gr(\mathsf{C})$*.*

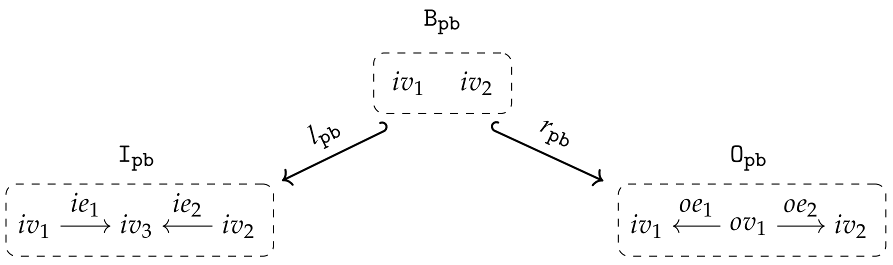

$\mathtt{B}_{\mathtt{pb}}$
iv_1 iv_2
l_{pb}
r_{pb}
$\mathtt{I}_{\mathtt{pb}}$
$iv_1 \xrightarrow{ie_1} iv_3 \xleftarrow{ie_2} iv_2$
$\mathtt{O}_{\mathtt{pb}}$
$iv_1 \xleftarrow{oe_1} ov_1 \xrightarrow{oe_2} iv_2$

Figure 3. Arity declaration for the operation symbol $\texttt{pb}$.

Remark 4.3 (Built-in Projections)**.** *Given a "set of indices"* I *and a "carrier set"* A*, we do have a projection map* $\pi_i : A^I \to A$ *at hand for any index* $i \in I$*, as described in Section 2.*

Analogously, we obtain for a "graph of indices" $\mathtt{H}$ *and a "carrier graph"* $\mathtt{G}$ *a projection map* $\pi^{\mathtt{H}}_{\mathtt{K}}$ *for any subgraph* $\mathtt{K}$ *of* $\mathtt{H}$ *by simple precomposition with the inclusion graph homomorphism* $\iota_{\mathtt{K},\mathtt{H}} : \mathtt{K} \hookrightarrow \mathtt{H}$*:*

$$\pi^{\mathtt{H}}_{\mathtt{K}} : \mathtt{G}^{\mathtt{H}} \to \mathtt{G}^{\mathtt{K}} \quad \text{is defined by} \quad \pi^{\mathtt{H}}_{\mathtt{K}}(\boldsymbol{b}) := \iota_{\mathtt{K},\mathtt{H}}\,;\boldsymbol{b} \quad \text{for all } \boldsymbol{b} \in \mathtt{G}^{\mathtt{H}}. \tag{17}$$

In this way, we do have for any Γ*-algebra* $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ *and any arity span* $\mathtt{I} \xhookleftarrow{l} \mathtt{B} \xlongequal{r} \mathtt{O}$ *(as in Definition 4.1) with* $\mathtt{B} = \mathtt{O}$ *exactly one map from* $\mathtt{G}^{\mathtt{I}}$ *to* $\mathtt{G}^{\mathtt{O}}$ *satisfying the commutativity condition for graph operations in Definition 4.2, namely the projection* $\pi^{\mathtt{I}}_{\mathtt{O}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$*. In case* $\mathtt{I} = \mathtt{B} = \mathtt{O}$*,* $\pi^{\mathtt{I}}_{\mathtt{I}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{I}}$ *is simply the identity on* $\mathtt{G}^{\mathtt{I}}$*.*

We call these projections built-in *since their semantics is completely determined by their arity! After the choice of the carrier* $\mathtt{G}$ *of a graph algebra* $\mathcal{G}$ *we do have these projections available independent of and prior to the choice of the semantics* $\texttt{op}^{\mathcal{G}}$ *of the operation symbols* $\texttt{op} \in OP$*.*

A crucial methodological point is that we can use these built-in projections without being forced to include corresponding auxiliary operational symbols in OP and/or without any need to define their semantics when defining graph algebras. In the traditional approach, we are forced to do this because there is no concept of boundaries at all, or, in other words, all boundaries in our sense are by default empty in traditional Universal Algebra.

Our reformulation of the definition of homomorphisms for traditional algebras in Definition 3.3 applies analogously to graph algebras.

Definition 4.3 (Graph Algebra Homomorphism)**.** *Let* Γ *be a graph signature. A* Γ*-homomorphism* $\varphi\colon \mathcal{G} \to \mathcal{H}$ *between two* Γ*-algebras* $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ *and* $\mathcal{H} = (\mathtt{H}, OP^{\mathcal{H}})$ *is a graph homomorphism* $\varphi\colon \mathtt{G} \to \mathtt{H}$ *satisfying the following homomorphism condition:*

$$\textbf{(HC)} \quad \texttt{op}^{\mathcal{G}}(\boldsymbol{b});\varphi = \texttt{op}^{\mathcal{H}}(\boldsymbol{b};\varphi) \quad \text{for all } \texttt{op} \in OP \text{ and all } \boldsymbol{b} \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}.$$

$$\begin{tikzcd}
\mathtt{I}_{\mathtt{op}} \arrow[r, "\boldsymbol{b}"] \arrow[dr, "\boldsymbol{b};\varphi"'] & \mathtt{G} \arrow[d, "\varphi"] & \mathtt{O}_{\mathtt{op}} \arrow[l, "\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})"'] \arrow[dl, "\mathtt{op}^{\mathcal{H}}(\boldsymbol{b};\varphi)"] \\
& \mathtt{H} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathtt{G}^{\mathtt{I}_{\mathtt{op}}} \arrow[r, "\mathtt{op}^{\mathcal{G}}"] \arrow[d, "\_;\varphi"'] \arrow[dr, phantom, "\circlearrowleft"] & \mathtt{G}^{\mathtt{O}_{\mathtt{op}}} \arrow[d, "\_;\varphi"] \\
\mathtt{H}^{\mathtt{I}_{\mathtt{op}}} \arrow[r, "\mathtt{op}^{\mathcal{H}}"] & \mathtt{H}^{\mathtt{O}_{\mathtt{op}}}
\end{tikzcd}$$

For any graphs $\mathtt{G}$, $\mathtt{H}$, and $\mathtt{J}$ each graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$ induces by post-composition a map $\_;\varphi : \mathtt{G}^{\mathtt{J}} \to \mathtt{H}^{\mathtt{J}}$ and thus we can, more abstractly but equivalently, express the homomorphism condition **(HC)** via the requirement that the above right square of maps commutes. Note that in the case of constant symbols $\mathtt{c} \in OP$, $\mathtt{I}_{\mathtt{c}} = \mathbf{0}$, the homomorphism condition turns into the equation $\mathtt{op}^{\mathtt{G}}((),()); \varphi = \mathtt{op}^{\mathtt{H}}((),())$ if we apply our conventions in Section 2 and Remark 4.2 concerning the tuple notation.

Example 4.4 (Functors as Homomorphisms). *In the case of Γ_{cat}-algebras as defined in Example 4.2, the homomorphism conditions for the operation symbols* `comp` *and* `id`*, according to Definition 3.3, are nothing but the usual requirements for functors to be compatible with composition and identities, respectively.*

Γ-algebras and Γ-homomorphisms together constitute a category $\mathsf{Alg}(\Gamma)$: The composition $\varphi;\psi : \mathcal{G} \to \mathcal{K}$ of two Γ-homomorphisms $\varphi : \mathcal{G} \to \mathcal{H}$ and $\psi : \mathcal{H} \to \mathcal{K}$ is given by the composition $\varphi;\psi$ of the underlying graph homomorphisms $\varphi : \mathtt{G} \to \mathtt{H}$ and $\psi : \mathtt{H} \to \mathtt{K}$. Lastly, the identity Γ-homomorphism $id_{\mathcal{G}} : \mathcal{G} \to \mathcal{G}$ for any Γ-algebra $\mathcal{G}$ is given by the identity graph homomorphism $id_{\mathtt{G}} : \mathtt{G} \to \mathtt{G}$.

Proposition 4.1 (Forgetful Functor). *The assignments $\mathcal{G} \to \mathtt{G}$ and $(\varphi : \mathcal{G} \to \mathcal{H}) \mapsto (\varphi : \mathtt{G} \to \mathtt{H})$ define a faithful forgetful functor:*

$$\mathbf{U}_{\Gamma} : \mathsf{Alg}(\Gamma) \to \mathsf{Graph}. \tag{18}$$

The homomorphism condition for Σ-homorphisms between Σ-algebras in Definition 3.3 has exactly the same structure as the homomorphism condition for Γ-homomorphisms between graph Γ-algebras in Definition 4.3. In such a way, the pure categorical proof of Theorem 3.1 can be directly transformed into a proof of the corresponding statement for graph algebras, and thus, we obtain the following theorem "for free".

Theorem 4.1 (Limits of Graph Algebras). $\mathsf{Alg}(\Gamma)$ *inherits any small limit from the category* Graph, *i.e., the functor* $\mathbf{U}_{\Gamma} : \mathsf{Alg}(\Gamma) \to \mathsf{Graph}$ creates *small limits.* $\mathsf{Alg}(\Gamma)$ *therefore has all small limits since* Graph *does.*

4.2. Comparison with the Old Definitions

Guided by the pioneering generalized sketch framework developed in the 1990s by a group around Zinovy Diskin [3–5], we introduced in [7] a different definition of graph signatures and graph algebras, respectively. Ref. [7] considers a graph inclusion $i_{\mathtt{op}} : \mathtt{I}_{\mathtt{op}} \hookrightarrow \mathtt{R}_{\mathtt{op}}$ as the arity of an operation symbol $\mathtt{op}$ and defines an operation $\mathtt{op}^{\mathcal{G}}$ on a graph $\mathtt{G}$ as a map from $\mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$ to $\mathtt{G}^{\mathtt{R}_{\mathtt{op}}}$, making the following triangle commute for any $\boldsymbol{b} : \mathtt{I}_{\mathtt{op}} \to \mathtt{G}$.

$$\begin{tikzcd}
\mathtt{I}_{\mathtt{op}} \arrow[rr, hook, "i_{\mathtt{op}}"] \arrow[dr, "\boldsymbol{b}"'] & \circlearrowleft & \mathtt{R}_{\mathtt{op}} \arrow[dl, "\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})"] \\
& \mathtt{G} &
\end{tikzcd} \tag{19}$$

The need for projection operations, among other issues, advised us to introduce explicit output arities. At that point, we could have also chosen cospans $\mathtt{I}_{\mathtt{op}} \hookrightarrow \mathtt{R}_{\mathtt{op}} \hookleftarrow \mathtt{O}_{\mathtt{op}}$

to declare the arities of graph operations instead of the spans in Definition 4.1. The choice of spans has, however, many advantages that we will try to point at later in the paper.

In this subsection, we argue that the new variant and the old variant in [7] are semantically equivalent as either definition of arity, algebra, or homomorphism can be obtained from the other.

4.2.1. Comparison of Arity Declarations

For any arity span $\mathtt{I}_{\mathrm{op}} \xleftarrow{l_{\mathrm{op}}} \mathtt{B}_{\mathrm{op}} \xrightarrow{r_{\mathrm{op}}} \mathtt{O}_{\mathrm{op}}$ in Definition 4.1, we can simply construct a pushout to obtain the *result arity* $\mathtt{R}_{\mathrm{op}}$

$$\begin{tikzcd}
 & \mathtt{B}_{\mathrm{op}} \arrow[dl, hook', "l_{\mathrm{op}}"'] \arrow[dr, hook, "r_{\mathrm{op}}"] & \\
\mathtt{I}_{\mathrm{op}} \arrow[dr, hook, "i_{\mathrm{op}}"] & po & \mathtt{O}_{\mathrm{op}} \arrow[dl, hook', "o_{\mathrm{op}}"'] \\
 & \mathtt{R}_{\mathrm{op}} &
\end{tikzcd}$$

The pushout of the arities of the operation symbol comp is visualized in Figure 4.

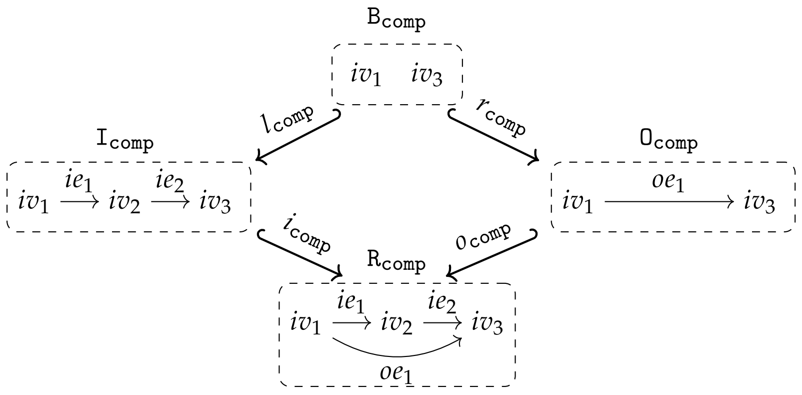

$\mathtt{B}_{\mathtt{comp}}$
iv_1 iv_3
$l_{\mathtt{comp}}$
$r_{\mathtt{comp}}$
$\mathtt{I}_{\mathtt{comp}}$
$iv_1 \xrightarrow{ie_1} iv_2 \xrightarrow{ie_2} iv_3$
$\mathtt{O}_{\mathtt{comp}}$
$iv_1 \xrightarrow{oe_1} iv_3$
$i_{\mathtt{comp}}$
$o_{\mathtt{comp}}$
$\mathtt{R}_{\mathtt{comp}}$
$iv_1 \xrightarrow{ie_1} iv_2 \xrightarrow{ie_2} iv_3$
oe_1

Figure 4. The pushout of arity declarations for the operation symbol comp.

Pushouts in Graph (as in any topos) preserve monomorphisms, and, moreover, pushouts with a monomorphism involved are also pullbacks ([16], 13.3). Equation (15) ensures that we can choose the specific pushout $\mathtt{R}_{\mathrm{op}} = \mathtt{I}_{\mathrm{op}} \cup \mathtt{O}_{\mathrm{op}}$, which makes the resulting i_{op} into a graph inclusion, matching the definition in [7]. Note that $\mathtt{B}_{\mathrm{op}} = \mathtt{O}_{\mathrm{op}}$ implies $\mathtt{I}_{\mathrm{op}} = \mathtt{R}_{\mathrm{op}}$.

Conversely, given a cospan $\mathtt{I}_{\mathrm{op}} \xhookrightarrow{i_{\mathrm{op}}} \mathtt{R}_{\mathrm{op}} \xhookleftarrow{o_{\mathrm{op}}} \mathtt{O}_{\mathrm{op}}$ of graph inclusions, we can construct the intersection $\mathtt{B}_{\mathrm{op}} = \mathtt{I}_{\mathrm{op}} \cap \mathtt{O}_{\mathrm{op}}$, i.e., the componentwise intersection of the vertex and edge sets. This is well-defined as both $\mathtt{I}_{\mathrm{op}}$ and $\mathtt{O}_{\mathrm{op}}$ are subgraphs of the same graph $\mathtt{R}_{\mathrm{op}}$. The resulting commutative square of inclusion graph homomorphisms is a pullback in Graph where the span $\mathtt{I}_{\mathrm{op}} \xhookleftarrow{l_{\mathrm{op}}} \mathtt{B}_{\mathrm{op}} \xhookrightarrow{r_{\mathrm{op}}} \mathtt{O}_{\mathrm{op}}$ satisfies the condition in Definition 4.1. In the case where $\mathtt{R}_{\mathrm{op}} = \mathtt{I}_{\mathrm{op}} \cup \mathtt{O}_{\mathrm{op}}$, the square is also a pushout!

Arities of graph operations are, however, defined in [7] by a single graph inclusion $i_{\mathrm{op}} : \mathtt{I}_{\mathrm{op}} \hookrightarrow \mathtt{R}_{\mathrm{op}}$ only. In this situation, we can construct a cospan $\mathtt{I}_{\mathrm{op}} \xhookrightarrow{i_{\mathrm{op}}} \mathtt{R}_{\mathrm{op}} \xhookleftarrow{o_{\mathrm{op}}} \mathtt{O}_{\mathrm{op}}$ of graph inclusion with $\mathtt{O}_{\mathrm{op}}$ the smallest subgraph of $\mathtt{R}_{\mathrm{op}}$ containing all vertices in $(\mathtt{R}_{\mathrm{op}})_V \setminus (\mathtt{I}_{\mathrm{op}})_V$ and all edges in $(\mathtt{R}_{\mathrm{op}})_E \setminus (\mathtt{I}_{\mathrm{op}})_E$. By construction, we have $\mathtt{R}_{\mathrm{op}} = \mathtt{I}_{\mathrm{op}} \cup \mathtt{O}_{\mathrm{op}}$. As for cospans, in general, $\mathtt{B}_{\mathrm{op}}$ is defined to be the graph $\mathtt{I}_{\mathrm{op}} \cap \mathtt{O}_{\mathrm{op}}$. However, the crucial observation is that $\mathtt{B}_{\mathrm{op}}$ will always be a *discrete graph*, i.e., a graph without edges! Note that this construction is a special case of the construction of so-called *initial pushouts* in Graph ([9], 6.1).

Since, for a pushout of arities, $\mathtt{B}_{\mathrm{op}} = \mathtt{O}_{\mathrm{op}}$ implies $\mathtt{I}_{\mathrm{op}} = \mathtt{R}_{\mathrm{op}}$, this means, especially, that the original definition of arities of graph operations in [7] does not allow us to consider *built-in projections* (see Remark 4.3) as legal graph operations. This was one of the main reasons that we introduced spans of graph inclusions as arities in this paper.

In conclusion, the span and cospan version are inverse to each other (at least in Graph) while the original version in [7] is a special case of the cospan version, which is less expressive than the other two versions. All three variants give us, however, a pushout of arities and inclusion graph homomorphisms at hand.

4.2.2. Equivalence of Graph Operations

Given a pushout of arities, we consider an arbitrary input $\boldsymbol{b} : \mathtt{I}_{\mathrm{op}} \to \mathtt{G}$.

$$\begin{tikzcd}
 & & \mathtt{B}_{\mathrm{op}} \arrow[dll, hook, "l_{\mathrm{op}}"'] \arrow[drr, hook, "r_{\mathrm{op}}"] & & \\
\mathtt{I}_{\mathrm{op}} \arrow[drr, hook, "i_{\mathrm{op}}"] \arrow[dddrr, bend right, "\boldsymbol{b}"'] & & po & & \mathtt{O}_{\mathrm{op}} \arrow[dll, hook', "o_{\mathrm{op}}"'] \arrow[dddll, dashed, bend left, "\boldsymbol{o}"] \\
 & & \mathtt{R}_{\mathrm{op}} \arrow[dd, dashed, "\boldsymbol{r}"] & & \\
 & \circlearrowleft & & \circlearrowleft & \\
 & & \mathtt{G} & &
\end{tikzcd}$$

For any $\boldsymbol{o} : \mathtt{O}_{\mathrm{op}} \to \mathtt{G}$ with $r_{\mathrm{op}}; \boldsymbol{o} = l_{\mathrm{op}}; \boldsymbol{b}$, there exists, due to the pushout property, a unique $\boldsymbol{r_o} : \mathtt{R}_{\mathrm{op}} \to \mathtt{G}$ with $i_{\mathrm{op}}; \boldsymbol{r_o} = \boldsymbol{b}$ and $o_{\mathrm{op}}; \boldsymbol{r_o} = \boldsymbol{o}$. Conversely, for any $\boldsymbol{r} : \mathtt{R}_{\mathrm{op}} \to \mathtt{G}$ with $i_{\mathrm{op}}; \boldsymbol{r} = \boldsymbol{b}$, we have, trivially, $l_{\mathrm{op}}; \boldsymbol{b} = r_{\mathrm{op}}; (o_{\mathrm{op}}; \boldsymbol{r})$.

The uniqueness of mediating morphisms ensures that the assignments $\boldsymbol{o} \mapsto \boldsymbol{r_o}$ and $\boldsymbol{r} \mapsto o_{\mathrm{op}}; \boldsymbol{r}$ are inverse to each other. This observation ensures that there is a one-to-one correspondence between maps from $\mathtt{G}^{\mathtt{I}_{\mathrm{op}}}$ to $\mathtt{G}^{\mathtt{R}_{\mathrm{op}}}$, satisfying commutativity condition (19), and maps from $\mathtt{G}^{\mathtt{I}_{\mathrm{op}}}$ to $\mathtt{G}^{\mathtt{O}_{\mathrm{op}}}$, satisfying the commutativity condition in Definition 4.2.

4.2.3. Equivalence of Homomorphism Conditions

By extending the *equivalence of graph operations*, the equivalence of the respective homomorphism conditions can also be straightforwardly shown utilizing the uniqueness of mediating morphisms for the pushout of arities, as the interested reader may check.

Remark 4.4 (Graph of a Graph Operation)**.** *For any map $f : A \to B$, its* graph *is usually defined as the binary relation $\{(a, f(a)) \mid a \in A\} \subseteq A \times B$. Often, the story is even turned, and maps are introduced as those binary relations $f \subseteq A \times B$ that are* left-total *and* right-unique.

Given a pushout of arities and a graph operation $\mathtt{op}^{\mathcal{G}} : \mathtt{G}^{\mathtt{I}_{\mathrm{op}}} \to \mathtt{G}^{\mathtt{O}_{\mathrm{op}}}$, we could, analogously, consider the set $\{\boldsymbol{r}_{\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})} \mid \boldsymbol{b} \in \mathtt{G}^{\mathtt{I}_{\mathrm{op}}}\} \subseteq \mathtt{G}^{\mathtt{R}_{\mathrm{op}}}$ as the graph of the graph operation $\mathtt{op}^{\mathcal{G}}$. Utilizing the projections from $\mathtt{G}^{\mathtt{R}_{\mathrm{op}}}$ into $\mathtt{G}^{\mathtt{I}_{\mathrm{op}}}$ and $\mathtt{G}^{\mathtt{O}_{\mathrm{op}}}$, we could even lift up properties like left-total *and* right-unique *to characterize those subsets of $\mathtt{G}^{\mathtt{R}_{\mathrm{op}}}$ that correspond to graph operations.*

This observation may be a basis for defining Skolemization *in* Logics of Statements in Context *[1] once we have integrated operations into those logics.*

$\mathtt{R}_{\mathrm{op}}$ also has another important role, which was probably one of the reasons that the original definition of sketch operations in [3–5] relied on graph inclusions $i_{\mathrm{op}} : \mathtt{I}_{\mathrm{op}} \hookrightarrow \mathtt{R}_{\mathrm{op}}$. $\mathtt{R}_{\mathrm{op}}$ is the only place where the input items of a graph operation and the new items, created by the graph operation, can be related. In such a way, we have to use $\mathtt{R}_{\mathrm{op}}$ if we want to describe and specify properties of the output of a graph operation that depend on properties of the input.

Example 4.5 (Graph of Pullback Operation)**.** *Figure 5 shows the pushout of arity declarations for the operation symbol* `pb`*. Constructing the graph of pullback operations, as described in Example 3, considers for any chosen pullback the whole pullback square, and not just the pullback span.*

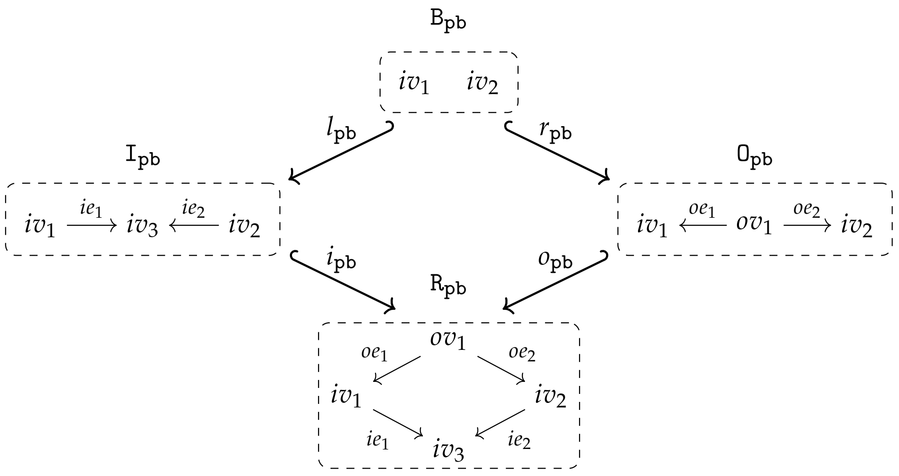


Figure 5. The pushout of arity declarations for the operation symbol `pb`.

Example 4.6 (Chosen (co)limits)**.** *In general, any chosen (co)limits of diagrams of a fixed shape* `I` *in a category* `C` *give rise to a corresponding graph operation on* $gr(\mathsf{C})$ *where the arity* `B` *is simply given by all the vertices in* `I` *while arity* `O` *represents the shape of the corresponding (co)cones.*

In such a way, the pushout `R` *combines the fixed shape* `I` *of diagrams with the shape of corresponding (co)cones, and thus, the elements in the graph of the corresponding graph operation on* $gr(\mathsf{C})$ *represent, at the same time, a diagram and a (co)cone for this diagram.*

Remark 4.5 (Advantages of Spans of Arities)**.** *Starting with a span of arities as in Definition 4.1, we obtain a corresponding commutative square of arities via a simple pushout construction. That this square becomes, moreover, in any topos, a pullback is an essential side effect.*

If we start, in contrast, with a cospan, we could construct a pullback to obtain a commutative square of arities. In the case of graphs (and probably in arbitrary pre-sheaf topoi), it is sufficient to require that the cospan of inclusion morphisms is jointly epic to make the pullback square simultaneously a pushout square. We are, however, not sure that this condition is sufficient for arbitrary topoi.

The tricky construction of initial pushouts *for a single arity inclusion (and not a cospan!) may also generalize to arbitrary pre-sheaf topoi but probably not to arbitrary topoi.*

The original definition of arities and graph operations in [7] turned out to not be appropriate to define derived graph operations*. On one hand, projections are necessary to define an appropriate notion of derived graph operations. The original definition excludes, however, projections. On the other hand, the construction of instances of graph operations in Section 5.3.2, by means of epi-mono factorizations and pushouts, cannot equivalently be mimicked by means of the original definition. Even the universal property of initial pushouts is not helpful in establishing an equivalence.*

4.3. Graph Subalgebras

The definitions and results for graph algebras presented in this subsection are new and not found in [7].

The effort we spent in Section 3 to lift up the traditional exposition of algebras to a more categorical one pays off now. We can directly transfer most of the definitions and results from algebras to graph algebras. The only difference is that the "ad hoc totalization trick" in the proof of Proposition 3.3 does not work in the case of graph algebras and that there is no *axiom of choice* in `Graph`.

In contrast to Section 3, we do not distinguish between “subalgebras” and “inclusion homomorphisms”. We define “subalgebras” simply as “inclusions”.

Definition 4.4 (Graph Subalgebra)**.** *Let $\Gamma = (OP, ar)$ be a graph signature. A Γ-algebra $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ is a* Γ-subalgebra *of a Γ-algebra $\mathcal{H} = (\mathtt{H}, OP^{\mathcal{H}})$ if $\mathtt{G} \sqsubseteq \mathtt{H}$ and and the inclusion graph homomorphism $\iota_{\mathtt{G},\mathtt{H}} = (\iota_{\mathtt{G}_V,\mathtt{H}_V}, \iota_{\mathtt{G}_E,\mathtt{H}_E}) : \mathtt{G} \to \mathtt{H}$ establishes a Γ-homomorphism $\iota_{\mathtt{G},\mathtt{H}} : \mathcal{G} \to \mathcal{H}$.*

We know that the monomorphisms (epimorphisms) in Graph are exactly the injective (surjective) graph homomorphisms, respectively. Since the functor $\mathbf{U}_\Gamma : \mathsf{Alg}(\Gamma) \to \mathsf{Graph}$ is faithful and faithful functors reflect monomorphisms and epimorphisms, we obtain

Corollary 4.1 (Injective and surjective Homomorphisms)**.** *If the underlying graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$ of a Γ-homomorphism $\varphi : \mathcal{G} \to \mathcal{H}$ is injective (surjective), then $\varphi : \mathcal{G} \to \mathcal{H}$ is a monomorphism (epimorphism) in $\mathsf{Alg}(\Gamma)$.*

The category Graph has all small limits and colimits, and those are obtained by componentwise limits and colimits, respectively, in Set. This means, especially, that Graph has all small *multiple pullbacks* (see Remark 3.3). Due to Theorem 4.1, we can define, in such a way, the intersection of graph subalgebras analogous to the intersection of subalgebras in Corollary 3.3.

Corollary 4.2 (Intersection of Graph Subalgebras)**.** *For any set I, any Γ-algebra $\mathcal{H}$, and any diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Alg}(\Gamma)$ of Γ-subalgebras $\delta_{e_i} = \iota_{\mathtt{G}_i,\mathtt{H}} : \mathcal{G}_i \hookrightarrow \mathcal{H}$, $i \in I$ of $\mathcal{H}$ there is a unique Γ-subalgebra $\mathcal{L} = (\mathtt{L}, OP^{\mathcal{L}})$ of $\mathcal{H}$ with $\mathtt{L} = \bigcap_{i\in I} \mathtt{G}_i$, i.e., $\mathtt{L}_V = \bigcap_{i\in I}(\mathtt{G}_i)_V$ and $\mathtt{L}_E = \bigcap_{i\in I}(\mathtt{G}_i)_E$, that is a Γ-subalgebra of $\mathcal{G}_i$ for all $i \in I$.*

Moreover, the inclusion Γ-homomorphisms $\iota_{\mathtt{L},\mathtt{G}_i} : \mathcal{L} \hookrightarrow \mathcal{G}_i$, $i \in I$ and $\iota_{\mathtt{L},\mathtt{H}} : \mathcal{L} \hookrightarrow \mathcal{H}$ constitute a multiple pullback, i.e., a limit cone of the diagram $\delta\colon \mathtt{MP}(I) \to \mathsf{Alg}(\Gamma)$.

We call $\mathcal{L} = (\mathtt{L}, OP^{\mathcal{L}})$ also the intersection *of the I-indexed family $\mathcal{M} = (\mathcal{G}_i \mid i \in I)$ of Γ-subalgebras of $\mathcal{H}$ and may use the notations $\bigcap \mathcal{M}$, $\bigcap_{i\in I} \mathcal{G}_i$ or, simply, $\bigcap \mathcal{G}_i$ to denote $\mathcal{L}$.*

The category Graph is *well powered*; i.e., the collection of all graph subalgebras of a graph algebra is a set, and thus, we can define the concept "accessible via a graph homomorphism".

Definition 4.5 (Graph Subalgebra accessible via a Graph Homomorphism)**.** *For any Γ-algebra $\mathcal{H}$ and any graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$, let $\mathcal{M}$ be the set of all Γ-subalgebras $\mathcal{X}$ of $\mathcal{H}$ such that φ factors through the inclusion graph homomorphism $\iota_{\mathtt{X},\mathtt{H}} : \mathtt{X} \hookrightarrow \mathtt{H}$; i.e., there exists a graph homomorphism $\varphi_{\mathtt{X}}\colon \mathtt{G} \to X$ such that $\varphi_{\mathcal{X}} \,;\iota_{\mathtt{X},\mathtt{H}} = \varphi$.*

We denote by $\mathcal{R}(\varphi, \mathcal{H})$ the intersection of $\mathcal{M}$, according to Corollary 4.2. In particular, the carrier of $\mathcal{R}(\varphi, \mathcal{H})$ is the intersection $\mathtt{R}(\varphi, \mathcal{H}) := \bigcap\{\mathtt{X} \mid \mathcal{X} \in \mathcal{M}\}$ of graphs. We call $\mathcal{R}(\varphi, \mathcal{H})$ the Γ-subalgebra of $\mathcal{H}$ accessible (reachable) via φ *or the* homomorphic image of $\mathtt{G}$ with respect to φ.

In the case of inclusion graph homomorphisms $\varphi = \iota_{\mathtt{G},\mathtt{H}}\colon \mathtt{G} \hookrightarrow \mathtt{H}$, we also use the notation $\mathcal{R}(\mathtt{G}, \mathcal{H})$ instead of $\mathcal{R}(\iota_{\mathtt{G},\mathtt{H}}, \mathcal{H})$ and also call $\mathcal{R}(\mathtt{G}, \mathcal{H})$ the Γ-subalgebra of $\mathcal{H}$ generated by $\mathtt{G}$.

Note that the graph homomorphism $\varphi_{\mathtt{X}}\colon \mathtt{G} \to X$ in Definition 4.5 is unique, if it exists, since the inclusion graph homomorphism $\iota_{X,B} : \mathtt{X} \hookrightarrow \mathtt{H}$ is a monomorphism in Graph.

We can also transfer Corollary 3.4 to graph algebras since, for any graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$, the set-theoretic image $\varphi(\mathtt{G})$ of $\mathtt{G}$ with respect to φ constitutes a subgraph of $\mathtt{H}$. Moreover, we have $\varphi(\mathtt{G}) = \bigcap\{\mathtt{Y} \mid \mathtt{Y} \in \mathcal{N}\}$ for the set $\mathcal{N}$ of all subgraphs $\mathtt{Y}$ of $\mathtt{H}$.

Corollary 4.3 (Homomorphic image includes Image)**.** *For any Γ-algebra $\mathcal{H}$ and any graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$, we have $\varphi(\mathtt{G}) \sqsubseteq \mathtt{R}(\varphi, \mathcal{H})$ for the set-theoretic image $\varphi(\mathtt{G})$ of $\mathtt{G}$ with respect to φ.*

Definition 4.6 (Accessible and Generated Graph Algebras)**.** *Let $\mathcal{H}$ be a Γ-algebra.*

1. *$\mathcal{H}$ is* accessible via a graph homomorphism *$\varphi : \mathtt{G} \to \mathtt{H}$ if $\mathcal{R}(\varphi, \mathcal{H}) = \mathcal{H}$.*
2. *If $\mathcal{H}$ is accessible via an inclusion graph homomorphism $\iota_{\mathtt{G},\mathtt{H}} : \mathtt{G} \to \mathtt{H}$, i.e., if $\mathcal{R}(\mathtt{G}, \mathcal{H}) = \mathcal{R}(\iota_{\mathtt{G},\mathtt{H}}, \mathcal{H}) = \mathcal{H}$, we say also that $\mathcal{H}$ is* generated by $\mathtt{G}$.
3. *$\mathcal{H}$ is said to be* generated *if it is generated by the empty graph, i.e., accessible via the unique graph homomorphism $\iota_{\mathbf{0},\mathtt{H}} : \mathbf{0} \hookrightarrow \mathtt{H}$ from the initial object $\mathbf{0}$ in* Graph *to* $\mathtt{H}$.

Corollary 4.4. *A Γ-algebra $\mathcal{H}$ is generated if, and only if, there are no proper Γ-subalgebras of $\mathcal{H}$.*

Corollary 4.5. *If a signature Γ has no constant symbols, then the empty Γ-algebra is the only generated Γ-algebra.*

The concept *accessible via a graph homomorphism* can be utilized to find a characterization of epimorphisms in $\mathsf{Alg}(\Gamma)$. First, we observe that "accessible" implies "epic".

Lemma 4.1 (Accessible implies Epic)**.** *A Γ-homomorphism $\varphi\colon \mathcal{G} \to \mathcal{H}$ is an epimorphism in* $\mathsf{Alg}(\Gamma)$ *if $\mathcal{H}$ is accessible via the underlying graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$, i.e., if $\mathcal{H} = \mathcal{R}(\varphi, \mathcal{H})$.*

Proof. We consider arbitrary Γ-homomorphsms $\psi, \phi : \mathcal{H} \to \mathcal{K}$ such that $\varphi; \psi = \varphi; \phi$.

We know that the subgraph $\mathtt{X}$ of $\mathtt{H}$ with $\mathtt{X}_V = \{v \in \mathtt{H}_V \mid \psi_V(v) = \phi_V(v)\} \subseteq \mathtt{H}_V$, $\mathtt{X}_E = \{e \in \mathtt{H}_E \mid \psi_E(e) = \phi_E(e)\} \subseteq \mathtt{H}_E$ together with the inclusion graph homomorphism $\iota_{\mathtt{X},\mathtt{H}} : \mathtt{X} \hookrightarrow \mathtt{H}$ is an equalizer of the graph homomorphisms $\psi, \phi : \mathtt{H} \to \mathtt{K}$ in Graph. According to Theorem 4.1, there is a unique Γ-algebra $\mathcal{X} = (\mathtt{X}, OP^{\mathcal{X}})$ such that $\iota_{\mathtt{X},\mathtt{H}} : \mathtt{X} \hookrightarrow \mathtt{H}$ becomes an inclusion Γ-homomorphism $\iota_{\mathtt{X},\mathtt{H}} : \mathcal{X} \hookrightarrow \mathcal{H}$, which is, moreover, the equalizer of the Γ-homomorphisms $\psi, \phi : \mathcal{H} \to \mathcal{K}$.

The assumption $\varphi; \psi = \varphi; \phi$ ensures that there exists a unique graph homomorphism $\varphi_{\mathcal{X}}\colon \mathtt{G} \to \mathtt{X}$ with $\varphi_{\mathcal{X}}\,; \iota_{\mathtt{X},\mathtt{H}} = \varphi$. Due to the construction of $\mathcal{R}(\varphi, \mathcal{H})$ in Definition 4.5, we have an inclusion Γ-homomorphism $\iota_{R(\varphi,\mathcal{H}),\mathtt{X}} : \mathcal{R}(\varphi, \mathcal{H}) \hookrightarrow \mathcal{X}$.

The accessibility of $\mathcal{H}$ means $\mathcal{H} = \mathcal{R}(\varphi, \mathcal{H})$, and thus, we obtain, finally, $\mathcal{X} = \mathcal{H}$. This means, however, that the equalizer of ψ and ϕ in Graph is the identity on $\mathtt{H}$ and thus, we have $\psi = \phi$ as required. □

We can also show that graph subalgebras are regular monomorphisms. Unfortunately, the "ad hoc totalization trick", used in the proof of Proposition 3.3, does not work for arbitrary graph operations, since we may have, in contrast to traditional operations, non-empty boundaries and a corresponding commutativity requirement for graph operations. The simplest example, where this trick fails, is an operation that simply outputs a chosen edge between two distinct nodes.

What we need is a more well-behaved procedure of transforming partial graph algebras into total graph algebras. We therefore develop in Section 4.4 a corresponding free construction called *term completion*.

Proposition 4.2 (Graph Subalgebras are Regular Monos)**.** *For any Γ-subalgebra $\iota_{\mathtt{G},\mathtt{H}}\colon \mathcal{G} \to \mathcal{H}$ of a Γ-algebra $\mathcal{H}$ there exists a Γ-algebra $\mathcal{K}$ and parallel Γ-homomorphisms $\psi, \phi\colon \mathcal{H} \to \mathcal{K}$ such that $\iota_{\mathtt{G},\mathtt{H}}\colon \mathcal{G} \to \mathcal{H}$ is the equalizer of ψ and ϕ in* $\mathsf{Alg}(\Gamma)$.

Proof. Utilizing the *term completion* construction and its characterization as a *free construction*, as presented in Section 4.4, we will sketch a proof modifying the proof of Proposition 3.3.

We construct the pushout of the span $\mathtt{H} \xleftarrow{\iota_{\mathtt{G},\mathtt{H}}} \mathtt{G} \xrightarrow{\iota_{\mathtt{G},\mathtt{H}}} \mathtt{H}$ of inclusion graph homomorphisms (see the left diagram below). We set $\mathtt{K} := \mathtt{H} +_{\mathtt{G}} \mathtt{H}$. Since $\iota_{\mathtt{G},\mathtt{H}}$ is monic in Graph both graph homomorphisms $\kappa_1, \kappa_2\colon \mathtt{H} \to \mathtt{K}$ are monic too and, moreover, the pushout square is as a pullback square as well. This ensures, especially, that $\iota_{\mathtt{G},\mathtt{H}} : \mathtt{G} \to \mathtt{H}$ is the equalizer of the graph homomorphisms $\kappa_1, \kappa_2 : \mathtt{H} \to \mathtt{K}$ in Graph.

Graph

$$\begin{tikzcd}
 & \mathtt{G} \arrow[dl, "\iota_{\mathtt{G},\mathtt{H}}"'] \arrow[dr, "\iota_{\mathtt{G},\mathtt{H}}"] & \\
\mathtt{H} \arrow[dr, "\kappa_1"'] & & \mathtt{H} \arrow[dl, "\kappa_2"] \\
 & \mathtt{K} &
\end{tikzcd}$$

$\mathsf{PAlg}(\Gamma)$

$$\begin{tikzcd}
 & \mathcal{G} \arrow[dl, "\iota_{\mathtt{G},\mathtt{H}}"'] \arrow[dr, "\iota_{\mathtt{G},\mathtt{H}}"] & \\
\mathcal{H} \arrow[dr, "\kappa_1"'] & & \mathcal{H} \arrow[dl, "\kappa_2"] \\
 & \mathcal{K} &
\end{tikzcd}$$

$\mathsf{Alg}(\Gamma)$

$$\begin{tikzcd}
 & \mathcal{G} \arrow[dl, "\iota_{\mathtt{G},\mathtt{H}}"'] \arrow[dr, "\iota_{\mathtt{G},\mathtt{H}}"] & \\
\mathcal{H} \arrow[r, "\kappa_1"] \arrow[dr, "\kappa_1;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}"'] & \mathcal{K} \arrow[d, "\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}"] & \mathcal{H} \arrow[l, "\kappa_2"'] \arrow[dl, "\kappa_2;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}"] \\
 & \mathcal{T}_\Gamma(\mathcal{K}) &
\end{tikzcd}$$

Graph Operations on $\mathtt{K}$: We extend $\mathtt{K}$ to a partial Γ-algebra $\mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}})$ (see Definition 4.7) by defining for each op in OP a corresponding partial graph operation $\mathsf{op}^{\mathcal{K}} : \mathtt{K}^{I_{\mathsf{op}}} \overset{\circ}{\rightarrow} \mathtt{K}^{O_{\mathsf{op}}}$ according to **Case 1**, **Case 2**, and **Case 3**.

The operations in $\mathcal{K}$ are defined exactly in a way that the graph homomorphisms κ_1 and κ_2 become Γ-homomorphisms $\kappa_1, \kappa_2 : \mathcal{H} \to \mathcal{K}$, in the sense of Definition 4.8, and thus we obtain a commutative square in PAlg (see the middle diagram above).

Applying the functor $\mathbf{TC}_\Gamma : \mathsf{PAlg}(\Gamma) \to \mathsf{Alg}(\Gamma)$, we transform this commutative square in $\mathsf{PAlg}(\Gamma)$ into a commutative square in $\mathsf{Alg}(\Gamma)$. Taking into account Corollary 4.6 and (23), we obtain the commutative diagram on the right above.

The underlying square of graph homomorphisms is again a pullback in Graph since the inclusion graph homomorphism $\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})} : \mathtt{K} \to \mathtt{T}_\Gamma(\mathcal{K})$ is monic in Graph.

Theorem 4.1 ensures, finally, that the Γ-homomorphism $\iota_{\mathtt{G},\mathtt{H}} : \mathcal{G} \to \mathcal{H}$ is the equalizer of the Γ-homomorphisms $\kappa_1;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}, \kappa_2;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})} : \mathcal{H} \to \mathcal{T}_\Gamma(\mathcal{K})$. □

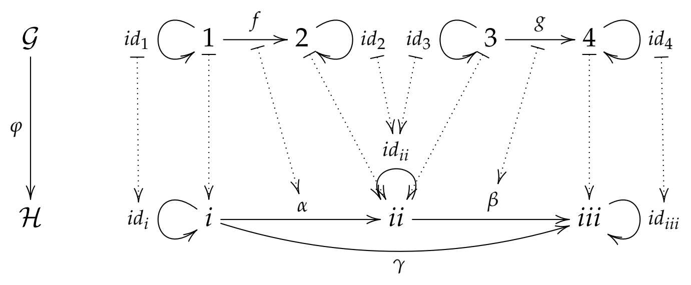
The regularity of $\mathsf{Alg}(\Gamma)$ entails that the concepts *accessible* and *epic* are equivalent.

Proposition 4.3 (Accessible $\cong$ Epic)**.** *For any Γ-homomorphism $\varphi : \mathcal{G} \to \mathcal{H}$, it holds that $\mathcal{H}$ is accessible via the underlying graph homomorphism $\varphi : \mathtt{G} \to \mathtt{H}$; i.e., in other words, $\mathcal{H}$ is equal to the homomorphic image $\mathcal{R}(\varphi, \mathcal{H})$ of $\mathcal{G}$ with respect to φ, if, and only if, $\varphi : \mathcal{G} \to \mathcal{H}$ is an epimorphism in* $\mathsf{Alg}(\Gamma)$.

Proof. "Accessible implies epic" has been shown in Lemma 4.1. We show now that "epic implies accessible": We consider an arbitrary Γ-subalgebra $\mathcal{X}$ of $\mathcal{H}$ such that there exists a graph homomorphism $\varphi_{\mathcal{X}} : \mathtt{G} \to \mathtt{X}$ with $\varphi_{\mathcal{X}} ; \iota_{\mathtt{X},\mathtt{B}} = \varphi$. Due to Proposition 4.2, there exist Γ-homomorphisms $\psi, \phi : \mathcal{H} \to \mathcal{C}$ such that $\iota_{\mathtt{X},\mathtt{H}} : \mathcal{X} \to \mathcal{H}$ is the equalizer of ψ and ϕ. Due to the assumption $\varphi_{\mathcal{X}} ; \iota_{\mathtt{X},\mathtt{H}} = \varphi$, we obtain $\varphi;\psi = \varphi;\phi$ and thus $\psi = \phi$, since φ is epic. This means, however, that $\mathcal{X} = \mathcal{H}$ and, finally, $\mathcal{R}(\varphi, \mathcal{H}) = \mathcal{H}$ due to the construction of $\mathcal{R}(\varphi, \mathcal{H})$ in Definition 4.5. □

The *axiom of choice* is not valid in Graph. Therefore, the set-theoretic image $\varphi(\mathtt{G})$ of the carrier $\mathtt{G}$ of a Γ-algebra $\mathcal{G}$ with respect to a Γ- homomorphism $\varphi : \mathcal{G} \to \mathcal{H}$ is, in general, not closed with respect to operations in $\mathcal{H}$. As a consequence, not every epic Γ-homomorphism needs to be surjective. We adapt the standard example of the composition of morphisms in categories.

Example 4.7 (Epic $\not\cong$ Surjective)**.** *We consider a Γ_{cat}-homomorphism $\varphi : \mathcal{G} \to \mathcal{H}$ between two finite Γ_{cat}-algebras $\mathcal{G}$ and $\mathcal{H}$, as depicted below.*



$\mathtt{G}^{\mathtt{I}_{\mathtt{comp}}}$ is empty, while $\mathtt{H}^{\mathtt{I}_{\mathtt{comp}}}$ has exactly one element $\boldsymbol{b}$ given by the assignments $f \mapsto \alpha$, $g \mapsto \beta$. α and β are in the set-theoretic image $\varphi(\mathtt{G})$, while the result of applying $\mathtt{comp}^{\mathcal{G}}$ to $\boldsymbol{b}$, namely $\gamma = \mathtt{comp}^{\mathcal{G}}(\boldsymbol{b})(oe_1)$, is not. $\varphi : \mathcal{G} \to \mathcal{H}$ is not surjective but epic since $\mathcal{H} = \mathcal{R}(\varphi, \mathcal{H})$.

4.4. Partial Graph Algebras and Their Term Completion

In practice, graph operations are often partial graph operations. The sketch operations introduced in [4,5], for example, can be seen as partial graph operations where the domain of definition is specified by diagrammatic predicates. Therefore, we have decided to also present in this paper the very basic definitions for partial graph algebras.

This decision was also influenced by the observation that we could prove Proposition 4.2 for arbitrary graph signatures based on a well-behaved completion procedure transforming partial (graph) algebras into total (graph) algebras. To our surprise, the construction of (graph) term algebras turns out to be just a special case of this new procedure.

Definition 4.7 (Partial Graph algebra)**.** *A* partial (graph) Γ-algebra *$\mathcal{G}$ is a pair $(\mathtt{G}, OP^{\mathcal{G}})$ given*

- *By a graph $\mathtt{G}$, called the* carrier *of $\mathcal{G}$, and*
- *By a family $OP^{\mathcal{G}} = (\mathtt{op}^{\mathcal{G}} \colon \mathtt{G}^{\mathtt{I}_{\mathtt{op}}} \not\to \mathtt{G}^{\mathtt{O}_{\mathtt{op}}} \mid \mathtt{op} \in OP)$ of partial maps such that the following diagram commutes for all $\mathtt{op}$ in OP and all graph homomorphisms $\boldsymbol{b} \in dom(\mathtt{G}^{\mathtt{I}_{\mathtt{op}}})$.*

$$\begin{tikzcd}
 & \mathtt{B}_{\mathtt{op}} \arrow[dl, hook', "l_{\mathtt{op}}"'] \arrow[dr, hook, "r_{\mathtt{op}}"] & \\
\mathtt{I}_{\mathtt{op}} \arrow[dr, "\boldsymbol{b}"'] & \circlearrowleft & \mathtt{O}_{\mathtt{op}} \arrow[dl, "\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})"] \\
 & \mathtt{G} &
\end{tikzcd} \tag{20}$$

The partial maps in $OP^{\mathcal{G}}$ are referred to as partial graph operations.

Be aware that constants can also be partial! For any constant symbol $\mathtt{c}$ in OP, with $\mathtt{I}_{\mathtt{c}} = \mathbf{0}$, we do have exactly two possibilities, since $\mathtt{G}^{\mathtt{I}_{\mathtt{c}}}$ is a singleton: Either $dom(\mathtt{c}^{\mathcal{G}}) = \mathtt{G}^{\mathtt{I}_{\mathtt{c}}}$, i.e., the constant is defined, or $dom(\mathtt{c}^{\mathcal{G}}) = \emptyset$, i.e., the constant is not defined.

Definition 4.8 (Partial Graph Algebra Homomorphism)**.** *A Γ-homomorphism $\varphi\colon \mathcal{G} \to \mathcal{H}$ between two partial Γ-algebras $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ and $\mathcal{H} = (\mathtt{H}, OP^{\mathcal{H}})$ is a graph homomorphism $\varphi\colon \mathtt{G} \to \mathtt{H}$ satisfying the following homomorphism condition*

(HCP) $\boldsymbol{b};\varphi \in dom(\mathtt{H}^{\mathtt{I}_{\mathtt{op}}})$ *and* $\mathtt{op}^{\mathcal{G}}(\boldsymbol{b});\varphi = \mathtt{op}^{\mathcal{H}}(\boldsymbol{b};\varphi)$ *for all* $\mathtt{op} \in OP$ *and all* $\boldsymbol{b} \in dom(\mathtt{G}^{\mathtt{I}_{\mathtt{op}}})$.

$$\begin{tikzcd}
\mathtt{I}_{\mathtt{op}} \arrow[r, "\boldsymbol{b}"] \arrow[dr, "\boldsymbol{b};\varphi"'] & \mathtt{G} \arrow[d, "\varphi"] \arrow[dl, phantom, "\circlearrowleft"] \arrow[dr, phantom, "\circlearrowleft"] & \mathtt{O}_{\mathtt{op}} \arrow[l, "\mathtt{op}^{\mathcal{G}}(\boldsymbol{b})"'] \arrow[dl, "\mathtt{op}^{\mathcal{H}}(\boldsymbol{b};\varphi)"] \\
 & \mathtt{H} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathtt{G}^{\mathtt{I}_{\mathtt{op}}} \arrow[r, "\mathtt{op}^{\mathcal{G}}"] \arrow[d, "\_;\varphi"'] \arrow[dr, phantom, "\geq"] & \mathtt{G}^{\mathtt{O}_{\mathtt{op}}} \arrow[d, "\_;\varphi"] \\
\mathtt{H}^{\mathtt{I}_{\mathtt{op}}} \arrow[r, "\mathtt{op}^{\mathcal{H}}"] & \mathtt{H}^{\mathtt{O}_{\mathtt{op}}}
\end{tikzcd}$$

In other words, the definedness of partial operations has to be preserved by a homomorphism but does not need to be reflected!

Partial graph Γ-algebras and Γ-homomorphisms between them constitute a category $\mathsf{PAlg}(\Gamma)$: The composition $\varphi;\psi : \mathcal{G} \to \mathcal{K}$ of two Γ-homomorphisms $\varphi : \mathcal{G} \to \mathcal{H}$ and $\psi : \mathcal{H} \to \mathcal{K}$ is given by the composition $\varphi;\psi$ of the underlying graph homomorphisms $\varphi : \mathtt{G} \to \mathtt{H}$ and $\psi : \mathtt{H} \to \mathtt{K}$. The identity Γ-homomorphism $id_{\mathcal{G}} : \mathcal{G} \to \mathcal{G}$ for any partial Γ-algebra $\mathcal{G}$ is given by the identity graph homomorphism $id_{\mathtt{G}} : \mathtt{G} \to \mathtt{G}$. We consider graph Γ-algebras as special partial Γ-algebras and thus $\mathsf{Alg}(\Gamma)$ is a full subcategory of $\mathsf{PAlg}(\Gamma)$.

Proposition 4.4 (Forgetful Functor)**.** *The assignments $\mathcal{G} \to \mathtt{G}$ and $(\varphi : \mathcal{G} \to \mathcal{H}) \mapsto (\varphi : \mathtt{G} \to \mathtt{H})$ define a faithful forgetful functor*

$$\mathbf{U}^{P}_{\Gamma} : \mathsf{PAlg}(\Gamma) \to \mathsf{Graph}.$$

By introducing a fresh new element whenever an operation is not defined for a certain input, we can transform any partial (graph) algebra into a total (graph) algebra.

Remark 4.6 (Syntactic Representation of Inputs)**.** *In addition to our notational conventions in Section 2 and Remark 4.2, we will rely on the following* syntactic representation *of inputs of graph operations: For any graph $\mathtt{G}$ and any $\mathtt{op}$ in OP an input $\boldsymbol{b} = (\boldsymbol{b}_V, \boldsymbol{b}_E) \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$ is represented by two strings, representing the vertices and the edges in $\mathtt{G}$, respectively, separated by the symbol "$\shortmid$"*

$$syn(\boldsymbol{b}) := \ulcorner bv_1, \ldots, bv_{n_{\mathtt{op}}} \shortmid be_1, \ldots, be_{m_{\mathtt{op}}} \urcorner$$

where $n_{\mathtt{op}} = |(\mathtt{I}_{\mathtt{op}})_V|$ and $m_{\mathtt{op}} = |(\mathtt{I}_{\mathtt{op}})_E|$. In the case $\mathtt{I}_{\mathtt{op}} = \mathbf{0}$, the only the input $\mathbf{0}_{\mathtt{G}} = ((),()) : \mathbf{0} \to \mathtt{G}$ is represented, in such a way, by two separated empty sequences: $syn(\mathbf{0}_{\mathtt{G}}) = \ulcorner \shortmid \urcorner$.

Of course, we could work with any other syntactic representation as long as the following two important properties are satisfied: (1) Uniqueness, i.e., for all $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$ we have $syn(\boldsymbol{b}_1) = syn(\boldsymbol{b}_2)$ if, and only if, $\boldsymbol{b}_1 = \boldsymbol{b}_2$; (2) $syn(\boldsymbol{b})$ is indeed a representation, i.e., we are able to reconstruct from $syn(\boldsymbol{b})$ the corresponding graph homomorphism $\boldsymbol{b} = (\boldsymbol{b}_V, \boldsymbol{b}_E) \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$ with the help of the information about $\mathtt{I}_{\mathtt{op}}$ in the signature Γ. In the case where $\mathtt{I}_{\mathtt{op}}$ has no isolated vertices, for example, we can uniquely represent any $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$ by the string $\ulcorner be_1, \ldots, be_{m_{\mathtt{op}}} \urcorner$ only!

Definition 4.9 (Term Completion)**.** *Let $\Gamma = (OP, ar)$ be a graph signature and $\mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}})$ be a partial Γ-algebra. We define the Γ-*term completion *$\mathtt{T}_{\Gamma}(\mathcal{K})$ of the graph $\mathtt{K}$ with respect to the partial Γ-algebra $\mathcal{K}$ as the smallest graph satisfying the following three conditions:*

Generators: $\mathtt{K} \sqsubseteq \mathtt{T}_{\Gamma}(\mathcal{K})$

Constants: *For all constants $\mathtt{c}$ in OP, such that $dom(\mathtt{c}^{\mathcal{K}}) = \emptyset$, the graph $\mathtt{T}_{\Gamma}(\mathcal{K})$ contains*

- $\ulcorner \mathtt{c}_{ov}\langle \shortmid \rangle \urcorner$ *as a vertex, for each vertex ov in $\mathtt{O}_{\mathtt{c}}$;*
- $\ulcorner \mathtt{c}_{oe}\langle \shortmid \rangle \urcorner$ *as an edge, for each edge oe in $\mathtt{O}_{\mathtt{c}}$,*
  where $sc^{\mathtt{T}_{\Gamma}(\mathcal{K})}(\mathtt{c}_{oe}\langle \shortmid \rangle) := \mathtt{c}_{sc^{\mathtt{O}_{\mathtt{c}}}(oe)}\langle \shortmid \rangle$ and $tg^{\mathtt{T}_{\Gamma}(\mathcal{K})}(\mathtt{c}_{oe}\langle \shortmid \rangle) := \mathtt{c}_{tg^{\mathtt{O}_{\mathtt{c}}}(oe)}\langle \shortmid \rangle$

Operations: *For all $\mathtt{op}$ in OP with $\mathtt{I}_{\mathtt{op}} \neq \mathbf{0}$ and any $\boldsymbol{t} \in \mathtt{T}_{\Gamma}(\mathcal{K})^{\mathtt{I}_{\mathtt{op}}}$ such that there is no $\boldsymbol{b} \in dom(\mathtt{op}^{\mathcal{K}}) \subseteq \mathtt{K}^{\mathtt{I}_{\mathtt{op}}}$ with $\boldsymbol{t} = \boldsymbol{b};\iota_{\mathtt{K},\mathtt{T}_{\Gamma}(\mathcal{K})}$, $\mathtt{T}_{\Gamma}(\mathcal{K})$ contains*

- $\ulcorner \mathtt{op}_{ov}\langle syn(\boldsymbol{t})\rangle \urcorner$ *as a vertex, for each vertex ov in $(\mathtt{O}_{\mathtt{op}})_V \setminus (\mathtt{B}_{\mathtt{op}})_V$;*

- $\ulcorner \mathtt{op}_{oe}\langle syn(\boldsymbol{t})\rangle \urcorner$ *as an edge, for each edge oe in* $(\mathtt{O_{op}})_E \setminus (\mathtt{B_{op}})_E$, *where*

$$sc^{\mathtt{T}_\Gamma(\mathcal{K})}(\mathtt{op}_{oe}\langle syn(\boldsymbol{t})\rangle) := \begin{cases} \boldsymbol{t}_V(sc^{\mathtt{O_{op}}}(oe)) & , \text{ if } sc^{\mathtt{O_{op}}}(oe) \in (\mathtt{B_{op}})_V \\ \mathtt{op}_{sc^{\mathtt{O_{op}}}(oe)}\langle syn(\boldsymbol{t})\rangle, & \text{if } sc^{\mathtt{O_{op}}}(oe) \in (\mathtt{O_{op}})_V \setminus (\mathtt{B_{op}})_V \end{cases}$$

$$tg^{\mathtt{T}_\Gamma(\mathcal{K})}(\mathtt{op}_{oe}\langle syn(\boldsymbol{t})\rangle) := \begin{cases} \boldsymbol{t}_V(tg^{\mathtt{O_{op}}}(oe)) & , \text{ if } tg^{\mathtt{O_{op}}}(oe) \in (\mathtt{B_{op}})_V \\ \mathtt{op}_{tg^{\mathtt{O_{op}}}(oe)}\langle syn(\boldsymbol{t})\rangle, & \text{if } tg^{\mathtt{O_{op}}}(oe) \in (\mathtt{O_{op}})_V \setminus (\mathtt{B_{op}})_V \end{cases}$$

Obviously, we have

$$\mathtt{T}_\Gamma(\mathcal{K}) = \mathtt{K} \quad \text{for all total } \Gamma\text{-algebras } \mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}}). \tag{21}$$

Remark 4.7 (Term Construction by Pushouts). *The construction of* Γ*-terms in Definition 4.9 can be organized as a successive application of term-construction steps starting with the graph* $\mathtt{K}$*: A* term-construction step *in the case* ***Constants*** *means that we construct, in parallel, for a constant symbol* $\mathtt{c}$ *the terms for all vertices and edges in* $\mathtt{O_c}$*. Analogously, a term-construction step in the case* ***Operations*** *means that we construct for an operation symbol* $\mathtt{op}$ *and an input* $\boldsymbol{t}$*, in parallel, the terms for all vertices and edges in* $\mathtt{O_{op}} \setminus \mathtt{B_{op}}$*. Each term-construction step extends an intermediate graph* $\mathtt{T}$ *to a graph* $\mathtt{T}'$*. This extension is, however, nothing but the construction of the following pushout:*

$$\begin{tikzcd} \mathtt{I_{op}} \arrow[d, "\boldsymbol{t}"'] & \mathtt{B_{op}} \arrow[l, hook', "l_{\mathtt{op}}"'] \arrow[r, hook, "r_{\mathtt{op}}"] \arrow[dl, "l_{\mathtt{op}};\boldsymbol{t}"] \arrow[d, phantom, "\ulcorner\cdot"] & \mathtt{O_{op}} \arrow[d, "\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t})"] \\ \mathtt{T} \arrow[rr, hook, "\iota_{\mathtt{T},\mathtt{T}'}"] \arrow[ur, phantom, "\circlearrowleft", near end] & & \mathtt{T}' \end{tikzcd}$$

where the definition of $\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t})$ *is spelled out, explicitly, in Definition 4.10. In light of this observation, one can look at the* term notation *as a means to solve two problems:*

1. *The term notation provides a uniform mechanism to create unique identifiers for the new graph items introduced by applying a non-deleting injective graph -transformation rule [9].*
2. *At the same time, the term notation encodes all the information necessary to uniquely identify the pushout that has been creating the new items.*

The following *term-completion construction* is new and has never even been defined for traditional partial algebras. Utilizing the operations in $\mathcal{K}$ to the greatest possible extent, we can straightforwardly extend $\mathcal{K}$ to a total Γ-algebra with carrier $\mathtt{T}_\Gamma(\mathcal{K})$.

Definition 4.10 (Term-Completion Algebra). *We can extend any partial* Γ*-algebra* $\mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}})$ *to a total* Γ*-algebra* $\mathcal{T}_\Gamma(\mathcal{K}) = (\mathtt{T}_\Gamma(\mathcal{K}), OP^{\mathcal{T}_\Gamma(\mathcal{K})})$ *as follows:*

Constants: *For all constants* $\mathtt{c}$ *in* OP*:*

Utilizing $\mathcal{K}$**:** *If* $\mathtt{c}^{\mathcal{K}}$ *is defined, we simply reuse it:*

$$\mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}) = \mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{K}};\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}) := \mathtt{c}^{\mathcal{K}}(\mathbf{0}_{\mathtt{K}});\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$$

Completion: *If* $\mathtt{c}^{\mathcal{K}}$ *is not defined, i.e.,* $dom(\mathtt{c}^{\mathcal{K}}) = \emptyset$ *we set*

- $(\mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}))_V(ov) := \ulcorner \mathtt{c}_{ov}\langle\,\mathsf{I}\,\rangle \urcorner$ *for each vertex ov in* $\mathtt{O_c}$ *and*
- $(\mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}))_E(oe) := \ulcorner \mathtt{c}_{oe}\langle\,\mathsf{I}\,\rangle \urcorner$ *for each edge oe in* $\mathtt{O_c}$.

Operations: *For all* $\mathtt{op}$ *in* OP *with* $\mathtt{I_{op}} \neq \mathbf{0}$ *and any* $\boldsymbol{t} \in \mathtt{T}_\Gamma(\mathcal{K})^{\mathtt{I_{op}}}$*:*

Utilizing $\mathcal{K}$**:** *If there is a* $\boldsymbol{b} \in dom(\mathtt{op}^{\mathcal{K}}) \subseteq \mathtt{K}^{\mathtt{I_{op}}}$ *with* $\boldsymbol{t} = \boldsymbol{b};\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$*, we reuse* $\mathtt{op}^{\mathcal{K}}$*:*

$$\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}) = \mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{b};\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}) := \mathtt{op}^{\mathcal{K}}(\boldsymbol{b});\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$$

Completion: *If there is no $\boldsymbol{b} \in dom(\mathtt{op}^{\mathcal{K}}) \subseteq \mathtt{K}^{\mathtt{I}_{\mathtt{op}}}$ with $\boldsymbol{t} = \boldsymbol{b}; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$, we set*

$$(\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}))_V(ov) := \begin{cases} \boldsymbol{t}_V(ov) & \text{, } \textit{if } ov \in (\mathtt{B}_{\mathtt{op}})_V \\ \ulcorner \mathtt{op}_{ov}\langle syn(\boldsymbol{t})\rangle \urcorner, & \textit{if } ov \in (\mathtt{O}_{\mathtt{op}})_V \setminus (\mathtt{B}_{\mathtt{op}})_V \end{cases}$$

$$(\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}))_E(oe) := \begin{cases} \boldsymbol{t}_E(oe) & \text{, } \textit{if } oe \in (\mathtt{B}_{\mathtt{op}})_E \\ \ulcorner \mathtt{op}_{oe}\langle syn(\boldsymbol{t})\rangle \urcorner, & \textit{if } oe \in (\mathtt{O}_{\mathtt{op}})_E \setminus (\mathtt{B}_{\mathtt{op}})_E. \end{cases}$$

The definitions ensure that the constructed pairs $\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t})$ of maps are indeed graph homomorphisms and that the operations in $\mathcal{T}_\Gamma(\mathcal{K})$ satisfy the commutativity condition in Definition 4.2. Moreover, the cases "**Utilizing** $\mathcal{K}$" are defined in such a way that we obtain

Corollary 4.6 (Embedding)**.** *For any partial Γ-algebra $\mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}})$, the inclusion graph homomorphism $\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})} : \mathtt{K} \to \mathtt{T}_\Gamma(\mathcal{K})$ constitutes a Γ-homomorphism $\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})} : \mathcal{K} \to \mathcal{T}_\Gamma(\mathcal{K})$ in* $\mathsf{PAlg}(\Gamma)$*, and thus, due to (21), $\mathcal{T}_\Gamma(\mathcal{K}) = \mathcal{K}$ if $\mathcal{K}$ is a total Γ-algebra.*

We can adapt and generalize the proof of Proposition 2 (Free graph algebras) in [7] to a proof that term completion is a free construction.

Proposition 4.5 (Term Completion as a Free Construction)**.** *For any partial Γ-algebra $\mathcal{K} = (\mathtt{K}, OP^{\mathcal{K}})$, the total Γ-algebra $\mathcal{T}_\Gamma(\mathcal{K}) = (\mathtt{T}_\Gamma(\mathcal{K}), OP^{\mathcal{T}_\Gamma(\mathcal{K})})$ has the following universal property: for any total Γ-algebra $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ and any Γ-homomorphism $\varphi\colon \mathcal{K} \to \mathcal{G}$ there exists a unique Γ-homomorphism $\varphi^*\colon \mathcal{T}_\Gamma(\mathcal{K}) \to \mathcal{G}$ such that the* defining condition $\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^* = \varphi$ *is satisfied.*

$$\begin{tikzcd}
\mathsf{PAlg}(\Gamma) & \mathcal{K} \arrow[r, hook, "\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}"] \arrow[dr, "\varphi"'] \arrow[dr, phantom, shift left=4, "\circlearrowleft"] & \mathcal{T}_\Gamma(\mathcal{K}) \arrow[d, "\varphi^*"] & \mathcal{T}_\Gamma(\mathcal{K}) \arrow[d, "\varphi^*"] & \mathsf{Alg}(\Gamma) \\
 & & \mathcal{G} & \mathcal{G} &
\end{tikzcd}$$

Proof. We prove this by structural induction according to Definition 4.9 and Remark 4.7.

Generators: In this base case, the defining condition forces $\varphi^*_V(kv) = \varphi_V(kv)$ for all $kv \in \mathtt{K}_V$ and $\varphi^*_E(kv) = \varphi_E(ke)$ for all $ke \in \mathtt{K}_E$.

Constants: In the second base case, we have, for all constants $\mathtt{c}$ in OP:

Utilizing $\mathcal{K}$: If $\mathtt{c}^{\mathcal{K}}$ is defined, the definition of operations in $\mathcal{T}_\Gamma(\mathcal{K})$, the defining condition and the assumption that φ is a Γ-homomorphism ensure that φ^* satisfies the homorphism condition for the constant $\mathtt{c}$:

$$\begin{aligned} \mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}); \varphi^* &= \mathtt{c}^{\mathcal{K}}(\mathbf{0}_{\mathtt{K}}); \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^* = \mathtt{c}^{\mathcal{K}}(\mathbf{0}_{\mathtt{K}}); \varphi = \mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{K}}; \varphi) \\ &= \mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{K}}; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^*) = \mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^*) \end{aligned}$$

Completion: If $\mathtt{c}^{\mathcal{K}}$ is not defined, the definition of operations in $\mathcal{T}_\Gamma(\mathcal{K})$ and the required homomorphism condition for φ^* forces for each vertex ov in $\mathtt{O}_{\mathtt{c}}$

$$\begin{aligned} \varphi^*_V(\mathtt{c}_{ov}\langle \shortmid \rangle) = \varphi^*_V\Big((\mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}))_V(ov)\Big) &= \Big(\mathtt{c}^{\mathcal{T}_\Gamma(\mathcal{K})}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}); \varphi^*\Big)_V(ov) \\ &= (\mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^*))_V(ov) = (\mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{G}}))_V(ov)\,. \end{aligned}$$

For each edge oe in $\mathtt{O}_{\mathtt{c}}$ we obtain, analogously, $\varphi^*_E(\mathtt{c}_{oe}\langle \shortmid \rangle) = (\mathtt{c}^{\mathcal{G}}(\mathbf{0}_{\mathtt{G}}))_E(oe)$.

Operations: We have, for all $\mathtt{op}$ in OP with $\mathtt{I}_{\mathtt{op}} \neq \mathbf{0}$ and any $\boldsymbol{t} \in \mathtt{T}_\Gamma(\mathcal{K})^{\mathtt{I}_{\mathtt{op}}}$,

Utilizing $\mathcal{K}$: If there is a $\boldsymbol{b} \in dom(\mathtt{op}^{\mathcal{K}}) \subseteq \mathtt{K}^{\mathtt{I}_{\mathtt{op}}}$ with $\boldsymbol{t} = \boldsymbol{b}; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$, then the definition of operations in $\mathcal{T}_\Gamma(\mathcal{K})$, the defining condition and the assumption that φ is a Γ-homomorphism ensure that φ^* satisfies the homomorphism condition for $\boldsymbol{t}$:

$$\begin{aligned}\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}); \varphi^* &= \mathtt{op}^{\mathcal{K}}(\boldsymbol{b}); \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^* = \mathtt{op}^{\mathcal{K}}(\boldsymbol{b}); \varphi = \mathtt{op}^{\mathcal{G}}(\boldsymbol{b}; \varphi) \\ &= \mathtt{op}^{\mathcal{G}}(\boldsymbol{b}; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}; \varphi^*) = \mathtt{op}^{\mathcal{G}}(\boldsymbol{t}; \varphi^*)\end{aligned}$$

Completion: If there is no $\boldsymbol{b} \in dom(\mathtt{op}^{\mathcal{K}}) \subseteq \mathtt{K}^{\mathtt{I}_{\mathtt{op}}}$ with $\boldsymbol{t} = \boldsymbol{b}; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}$, the induction hypothesis is that φ^* is already defined on a subgraph $\mathtt{T} \sqsubseteq \mathtt{T}_\Gamma(\mathcal{K})$ and that $\boldsymbol{t}(\mathtt{I}_{\mathtt{op}}) \sqsubseteq \mathtt{T}$. This ensures $\boldsymbol{t}; \varphi^* \in \mathtt{G}^{\mathtt{I}_{\mathtt{op}}}$. In the induction step, we extend φ^* to the graph $\mathtt{T}' \sqsubseteq \mathtt{T}_\Gamma(\mathcal{K})$. The definition of operations in $\mathcal{T}_\Gamma(\mathcal{K})$ and the required homomorphism condition for φ^* forces for each vertex ov in $(\mathtt{O}_{\mathtt{op}})_V \setminus (\mathtt{B}_{\mathtt{op}})_V$

$$\begin{aligned}\varphi^*_V(\mathtt{op}_{ov}\langle syn(\boldsymbol{t})\rangle) &= \varphi^*_V\Big((\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}))_V(ov)\Big) = \Big(\mathtt{op}^{\mathcal{T}_\Gamma(\mathcal{K})}(\boldsymbol{t}); \varphi^*\Big)_V(ov) \\ &= (\mathtt{op}^{\mathcal{G}}(\boldsymbol{t}; \varphi^*))_V(ov)\,.\end{aligned}$$

For each edge oe in $(\mathtt{O}_{\mathtt{op}})_E \setminus (\mathtt{B}_{\mathtt{op}})_E$ we obtain, analogously, $\varphi^*_E(\mathtt{op}_{oe}\langle syn(\boldsymbol{t})\rangle) = (\mathtt{op}^{\mathcal{G}}(\boldsymbol{t}; \varphi^*))_E(oe)$.

□

The assignments $\mathcal{K} \mapsto \mathcal{T}_\Gamma(\mathcal{K})$ and $(\psi : \mathcal{L} \to \mathcal{K}) \mapsto ((\psi; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})})^* : \mathcal{T}_\Gamma(\mathcal{L}) \to \mathcal{T}_\Gamma(\mathcal{K}))$ define, as usual for free constructions, a functor $\mathbf{TC}_\Gamma : \mathsf{PAlg}(\Gamma) \to \mathsf{Alg}(\Gamma)$, and this functor is left-adjoint to the inclusion functor $\mathbf{I} : \mathsf{Alg}(\Gamma) \hookrightarrow \mathsf{PAlg}(\Gamma)$.

$$\begin{tikzcd} & \mathsf{PAlg}(\Gamma) \arrow[r, shift left, "\mathbf{TC}_\Gamma"] & \mathsf{Alg}(\Gamma) \arrow[l, hook', shift left, "\mathbf{I}"] & \\ \mathcal{L} \arrow[r, "\iota_{\mathtt{L},\mathtt{T}_\Gamma(\mathcal{L})}"] \arrow[d, "\psi"'] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{T}_\Gamma(\mathcal{L}) \arrow[d, "(\psi;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})})^*"] & \mathcal{T}_\Gamma(\mathcal{L}) \arrow[d, "(\psi;\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})})^*"] \\ \mathcal{K} \arrow[r, "\iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})}"] & \mathcal{T}_\Gamma(\mathcal{K}) & \mathcal{T}_\Gamma(\mathcal{K}) \end{tikzcd} \tag{22}$$

Due to Corollary 4.6 we even have $\mathbf{I}; \mathbf{TC}_\Gamma = id_{\mathsf{Alg}(\Gamma)}$, i.e., $\mathsf{Alg}(\Gamma)$ is a *full reflective subcategory* of $\mathsf{PAlg}(\Gamma)$. In particular, we have for any Γ-homomorphism $\psi : \mathcal{L} \to \mathcal{K}$ in $\mathsf{PAlg}(\Gamma)$

$$\mathbf{TC}_\Gamma(\psi) = (\psi; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})})^* = \psi; \iota_{\mathtt{K},\mathtt{T}_\Gamma(\mathcal{K})} \quad \text{if } \mathcal{L} \text{ is a total } \Gamma\text{-algebra.} \tag{23}$$

4.5. Graph Terms and Graph Term Algebras

Graph term algebras are just the special case of term-completion algebras $\mathcal{T}_\Gamma(\mathcal{K})$ where all the graph operations in $\mathcal{K}$ are completely undefined, i.e., where everything is determined by the carrier only.

Definition 4.11 (Graph Term Algebras)**.** *Let* $\mathtt{X}$ *be a graph and* $\mathcal{X} = (\mathtt{X}, OP^{\mathcal{X}})$ *be the corresponding unique partial* Γ*-algebra where all graph operations are completely undefined.*

Graph terms *: We denote the graph* $\mathtt{T}_\Gamma(\mathcal{X})$*, according to Definition 4.9, also by* $\mathtt{T}_\Gamma(\mathtt{X})$ *and call it the* graph of all (graph) Γ-terms on $\mathtt{X}$.

Graph term algebra *: We denote the term completion* Γ*-algebra* $\mathcal{T}_\Gamma(\mathcal{X})$*, due to Definition 4.10, also by* $\mathcal{T}_\Gamma(\mathtt{X}) = (\mathtt{T}_\Gamma(\mathtt{X}), OP^{\mathcal{T}_\Gamma(\mathtt{X})})$ *and call it the* Γ-term graph algebra on $\mathtt{X}$.

Assigning to any graph $\mathtt{X}$ the corresponding unique partial Γ-algebra $\mathcal{X} = (\mathtt{X}, OP^{\mathcal{X}})$ where all graph operations are completely undefined, defines a functor from Graph to $\mathsf{PAlg}(\Gamma)$ that is left-adjoint to the forgetful functor $\mathbf{U}^P_\Gamma : \mathsf{PAlg}(\Gamma) \to \mathsf{Graph}$ in Proposition 4.4.

Combining this adjunction with the adjunction in (22) gives us the desired universal property of graph term algebras at hand.

Proposition 4.6 (Graph Term Algebra as a Free Construction)**.** *Given a graph* $\mathtt{X}$*, the* Γ*-term graph algebra* $\mathcal{T}_\Gamma(\mathtt{X})$ *has the following universal property: For any total* Γ*-algebra* $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ *and any graph homomorphism* $\varphi\colon \mathtt{X} \to \mathtt{G}$*, there exists a unique* Γ*-homomorphism* $\varphi^*\colon \mathcal{T}_\Gamma(\mathtt{X}) \to \mathcal{G}$ *such that the* defining condition $\iota_{\mathtt{X},\mathtt{T}_\Gamma(\mathtt{X})};\varphi^* = \varphi$ *is satisfied.*

$$\begin{tikzcd}
\mathsf{Graph} & \mathtt{X} \arrow[r, hook, "\iota_{\mathtt{X},\mathtt{T}_\Gamma(\mathtt{X})}"] \arrow[dr, "\varphi"'] \arrow[dr, phantom, shift left=4, "\circlearrowleft"] & \mathtt{T}_\Gamma(\mathtt{X}) \arrow[d, "\varphi^*"] & \mathcal{T}_\Gamma(\mathtt{X}) \arrow[d, "\varphi^*"] & \mathsf{Alg}(\Gamma) \\
& & \mathtt{G} & \mathcal{G} &
\end{tikzcd}$$

It might be worth mentioning that Proposition 4.6 also enables us to straightforwardly transfer Lemmas 3.3 and 3.4 in Section 3.5 to the graph algebra setting.

The definition of graph term algebras and their characterization as a free construction is the main result in [7]. We claimed, "The Kleisli category of the new adjunction provides an appropriate substitution calculus." However, as time passed, we realized that this claim is not fully true.

Substition Calculus: Graph term algebras manifest the "internalization approach" in the case of graph algebras. Relying on Proposition 4.6, we can indeed obtain a fully fledged substitution calculus, meeting the requirements formulated in Section 3.4.1. Based on the idea that a *substitution (declaration)* is now given by a graph homomorphism $\sigma : \mathtt{X} \to \mathtt{T}_\Gamma(\mathtt{Y})$ and that a *variable assignment* is a graph homomorphism $\alpha : \mathtt{X} \to \mathtt{G}$ for a Γ-algebra $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$, we can simply transfer all the discussions, definitions and results from Section 3.4.2 to graph algebras. We will spare the reader this copy–paste exercise.

No appropriate concept of Derived Operation: In traditional Universal Algebra, we do have a one-to-one correspondence between the "internal view" of terms as elements of free algebras and the "external view" of terms as an appropriate representation of *derived operations* (compare Definitions 3.12 and 3.14). It took us a while to realize that this one-to-one correspondence breaks down when it comes to graph algebras and even longer to understand the reasons. We discuss and address this problem in the next section.

5. Derived Graph Operations

We now discuss the reasons why graph terms, in our opinion, do not provide a fully adequate and appropriate concept of *derived graph operation*. First, each graph term, interpreted as an operation in a given graph algebra, will only produce isolated single vertices or single edges (without the source or target, respectively). What we do need, however, are graphs as outputs of derived graph operations! This flaw could be repaired by considering not single graph terms but subgraphs of graphs of graph terms.

Lemma 5.1 (Operations by Subgraphs)**.** *For a given graph* $\mathtt{X}$*, any subgraph* $\mathtt{O} \sqsubseteq \mathtt{T}_\Gamma(\mathtt{X})$ *defines a span* $\mathtt{X} \overset{l_\mathtt{O}}{\hookleftarrow} \mathtt{B}_\mathtt{O} \overset{o_\mathtt{O}}{\hookrightarrow} \mathtt{O}$ *of graph inclusions with* $\mathtt{B}_\mathtt{O} := \mathtt{X} \cap \mathtt{O}$*. Moreover, we obtain for all graphs* $\mathtt{G}$ *a map* $\delta^{\mathtt{G}}_{\mathtt{O}} : \mathtt{G}^{\mathtt{X}} \to \mathtt{G}^{\mathtt{O}}$ *defined by* $\delta^{\mathtt{G}}_{\mathtt{O}}(\boldsymbol{b}) := \iota_{\mathtt{O},\mathtt{T}_\Gamma(\mathtt{X})};\boldsymbol{b}^*$ *for all* $\boldsymbol{b} \in \mathtt{G}^{\mathtt{X}}$ *and satisfying the commutativity requirement for graph operations in Definition 4.2.*

However, this solution is not quite satisfactory. Each item in $\mathtt{O}$ is given by a separate term expression, and the different term expressions may represent, in general, different "computation schemes". What we want and need, especially in practical applications, is a single *graph operation expression* built up from variables and the symbols in OP such that the corresponding derived operation for a Γ-algebra produces, for any input, all (!) output items simultaneously with the same computation. Moreover, we would like to be able to define the semantics of these graph operation expressions, i.e., the corresponding *derived graph operations* in Γ-algebras, independently of graph term algebras and in a comparably easy, well-structured inductive way as we did for terms in Definition 3.14.

Unfortunately, traditional terms do not provide a fully appropriate blueprint to define such graph operation expressions. In Definition 3.8, terms are constructed using the following steps: The two basic steps, **Variables** and **Constants**, and the induction step **Operations**, which is implicitly split into two steps—(1) **Tupling** and (2) **Symbolic Sequential Composition** of a tuple with an operation symbol. This splitting becomes apparent in Definition 3.14 (Construction of Derived Operations).

In the case of graph operations, the step **Variables** turns into a step **(Built-in) Projections**. Besides this, there is nothing wrong with any of the steps except the step **Tupling**.

Example 5.1 (Composition of four Edges)**.** *To illustrate the problems with tupling, we consider the composition of four edges. We are interested in a "graph operation expression" built up of three copies of the operation symbol* `comp`*, as defined in Example 4.1. The input arity of the expression should be the graph* $iv_1 \xrightarrow{ie_1} iv_2 \xrightarrow{ie_2} iv_3 \xrightarrow{ie_3} iv_4 \xrightarrow{ie_4} iv_5$ *and the output arity should be the graph* $iv_1 \xrightarrow{oe} iv_5$ *with oe representing the composition of the four edges in the input arity.*

An obvious idea is that in a first step, two parallel applications of `comp` *produce the graph* $iv_1 \xrightarrow{oe_1} iv_3 \xrightarrow{oe_2} iv_5$ *with* oe_1 *representing the composition of the edges* ie_1*,* ie_2 *and* oe_2 *representing the composition of the edges* ie_3*,* ie_4*, respectively. In a second final step, the third application of* `comp` *should then produce the edge* $iv_1 \xrightarrow{oe} iv_5$ *.*

If we describe the first step by a tuple, we will not obtain $iv_1 \xrightarrow{oe_1} iv_3 \xrightarrow{oe_2} iv_5$ *as the output arity but only a pair* $\left(iv_1 \xrightarrow{oe_1} iv_3 \ , \ iv_3 \xrightarrow{oe_2} iv_5 \right)$ *of separated edges. This pair of edges, however, does not match the input arity of* `comp` *and thus the second step can not be performed.*

One could argue that we can repair this flaw a posteriori *by "gluing" the two separated graphs on the overlapping part, i.e., on the vertex* iv_3 *in the example. This construction, however, would be rather complicated and pathological, as it would consist of a mixture of colimit and limit constructions. In the next subsections, we will propose a more systematic and well-behaved mechanism based on* a priori *"soldering".*

5.1. Reconstruction of Syntactic Lawvere Theories

In this section, we analyze the construction of syntactic Lawvere theories in more detail to better understand the "nature of tupling" and to find a way to solve the problems with tupling pointed at in Example 5.1.

In Section 3.7, we defined syntactic Lawvere categories $\mathsf{L}(\Sigma)$, relying on a given concept of term and a corresponding substitution calculus. Moreover, we have seen that a syntactic Lawvere category $\mathsf{L}(\Sigma)$ can be characterized as a finite product category freely generated by a signature Σ. Following this observation, we will now turn the story and reconstruct the concept of term by means of the language of finite products and a corresponding axiomatization of finite products.

5.1.1. Categories with Finite Products

We start with a standard definition of finite products. A category C has finite products if, and only if, the following ingredients are present:

1. C has an *empty product (terminal object)* $\mathbf{1}$, i.e., for any object A in C there is a morphism $\langle\!\langle\, \rangle\!\rangle : A \to \mathbf{1}$ such that

$$g; \langle\!\langle\, \rangle\!\rangle = \langle\!\langle\, \rangle\!\rangle : B \to \mathbf{1} \quad \text{for all morphisms } g : B \to A. \tag{24}$$

2. For any family $A_1, \dots, A_n$, $n \geq 1$ of objects, there is an object $A_1 \times \dots \times A_n$ together with *projections* $\pi_i : A_1 \times \dots \times A_n \to A_i$, $1 \leq i \leq n$ such that
3. For any object B and any family $f_i : B \to A_i$, $1 \leq i \leq n$ of morphisms, there is a morphism $\langle\!\langle f_1, \dots, f_n \rangle\!\rangle : B \to A_1 \times \dots \times A_n$ with

$$\langle\!\langle f_1, \dots, f_n \rangle\!\rangle ; \pi_i = f_i \quad \text{for all } 1 \leq i \leq n. \tag{25}$$

   Moreover, we have $id_{\mathbf{1}} = \langle\!\langle\,\rangle\!\rangle$ and $id_{A_1 \times \dots \times A_n} = \langle\!\langle \pi_1, \dots, \pi_n \rangle\!\rangle$.
4. Finally, for all morphisms $g : C \to B$, the following equation holds

$$g ; \langle\!\langle f_1, \dots, f_n \rangle\!\rangle = \langle\!\langle g ; f_1, \dots, g ; f_n \rangle\!\rangle\,. \tag{26}$$

5.1.2. Categories Based on Finite Product Expressions

To reconstruct syntactic Lawvere categories, we define, in a first step, reflexive graphs with finite product expressions as edges and an associative composition. A *finite product expression* (or *fp-expression* for short) is a string of symbols built up of variable symbols, operation symbols, and angle bracket symbols "$\langle$", "$\rangle$" to denote tupling, the semicolon symbol " ; " to denote *symbolic composition*, and the auxiliary comma symbol " , " to separate substrings.

In the second step, we generate from fp-expression graphs finite product categories with equivalence classes of fp-expressions as morphisms.

We will only outline the definitions, constructions and results. One possibility to do it completely formally and precisely is to reuse, for example, the well-developed theory of specifications of partial algebras with conditional existence equations [17,18] and to construct the finite product categories as "partial quotient term algebras" freely generated by signatures (compare [19]).

First, we define for any signature $\Sigma = (OP, ar)$ a reflexive graph $\mathtt{FP}(\Sigma)$ with an associative composition as follows:

Objects: As objects, we choose the same canonical finite sets of variables as for $\mathsf{L}(\Sigma)$

$$X_0 = \varnothing \quad \text{and} \quad X_n := \{x^n_1, x^n_2, \dots, x^n_n\} \text{ for all } n \in \mathbb{N} \text{ with } n \geq 1\,.$$

Morphisms: Morphisms are all *finite product expressions* inductively defined as follows

Symbolic Projections: $\ulcorner x^n_i \urcorner : X_n \to X_1$ is an fp-expression for all $n \geq 1$, $1 \leq i \leq n$.

Constant and Operation Symbols: $\ulcorner op \urcorner : X_n \to X_1$ with $n \in \mathbb{N}$ is an fp-expression if op is an n-ary operation symbol in OP.

Empty Symbolic Tuples: $\ulcorner \langle \rangle \urcorner : X_n \to X_0$ is an fp-expression for all $n \in \mathbb{N}$.

Non-empty Symbolic Tuples: $\ulcorner \langle ex_1, \dots, ex_n \rangle \urcorner : X_m \to X_n$ is an fp-expression for all $m \geq 0$, $n \geq 1$ and all families $\ulcorner ex_i \urcorner : X_m \to X_1$, $1 \leq i \leq n$ of fp-expressions.

Symbolic Sequential Composition: $\ulcorner ex_1 ; ex_2 \urcorner : X_n \to X_m$ is an fp-expression for all $n, k, m \in \mathbb{N}$ and all fp-expressions $\ulcorner ex_1 \urcorner : X_n \to X_k$, $\ulcorner ex_2 \urcorner : X_k \to X_m$.

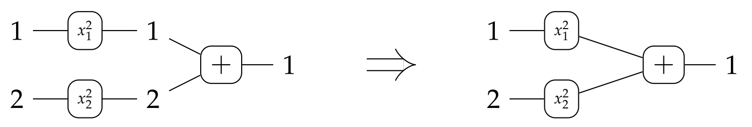
Symbolic Identities: $\ulcorner \langle x^n_1, \dots, x^n_n \rangle \urcorner : X_n \to X_n$ is the identity on X_n for all $n \geq 1$ and $\ulcorner \langle\,\rangle \urcorner : X_0 \to X_0$ is the identity on X_0.

Remark 5.1 (Computation Diagrams)**.** *Inspired by* logic circuit diagrams, term graphs *[20], and* string diagrams *[21], we will use informal* computation diagrams *to visualize the computations represented by fp-expressions. A computation diagram consists of "computation units", "(data-flow) edges", and input and output "ports".*

Each n-ary operation symbol is seen as a "computation unit" with n input ports and a single output port. Variable symbols appear, however, in two different roles: as "ports", i.e., as elements of the objects X_n, and as "computation units", i.e., as identifiers for projections. To distinguish these two roles, we simply denote ports by i instead of x^n_i. As a "computation unit", a variable simply copies values from a single input port to an arbitrary finite number of output ports.

Example 5.2 (Finite Product Expressions). *Let Σ be a signature with two binary operation symbols, "+" and "$*$". Both fp-expressions $\ulcorner + \urcorner : X_2 \to X_1$ and $\ulcorner \langle x_1^2, x_2^2 \rangle ; + \urcorner : X_2 \to X_1$ are equivalent, according to the Equations (24)–(26), and represent simple "tree-like" computation diagrams as depicted below. The picture below also visualizes the effect of symbolic composition.*

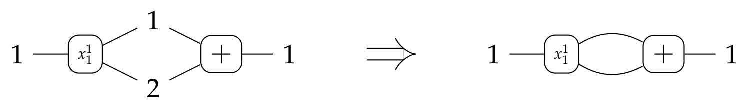
$$\begin{tikzcd} X_2 \arrow[r, "\langle x_1^2, x_2^2 \rangle"] & X_2 \arrow[r, "+"] & X_1 \end{tikzcd} \qquad \begin{tikzcd} X_2 \arrow[rr, "\langle x_1^2, x_2^2 \rangle ; +"] & & X_1 \end{tikzcd}$$



Equations (24)–(26) do not enforce that the unary symbolic tuple $\ulcorner \langle \langle x_1^2, x_2^2 \rangle ; + \rangle \urcorner : X_2 \to X_1$ is equivalent to the fp-expression $\ulcorner \langle x_1^2, x_2^2 \rangle ; + \urcorner : X_2 \to X_1$ even if both fp-expressions represent "essentially" the same computation and are therefore depicted by the same computation diagram!

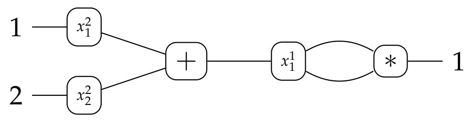
The first fp-expressions represent the addition of two numbers. We can, of course, derive expressions representing the doubling of a number as the expression $\ulcorner \langle x_1^1, x_1^1 \rangle ; + \urcorner : X_1 \to X_1$, for example. The picture below shows how the corresponding dag-like computation diagram is obtained by composing a "copying unit" with a computation unit.

$$\begin{tikzcd} X_1 \arrow[r, "\langle x_1^1, x_1^1 \rangle"] & X_2 \arrow[r, "+"] & X_1 \end{tikzcd} \qquad \begin{tikzcd} X_1 \arrow[rr, "\langle x_1^1, x_1^1 \rangle ; +"] & & X_1 \end{tikzcd}$$



More generally, fp-expressions allow us to represent arbitrary "sharing of sub-computations". The fp-expression $\ulcorner \langle \langle x_1^2, x_2^2 \rangle ; + \rangle ; \langle x_1^1, x_1^1 \rangle ; * \urcorner : X_2 \to X_1$, for example, represents the square of the sum of two numbers. The corresponding dag-like computation diagram is depicted below

$$\begin{tikzcd} X_2 \arrow[rrr, "\langle \langle x_1^2, x_2^2 \rangle ; + \rangle ; \langle x_1^1, x_1^1 \rangle ; *"] & & & X_1 \end{tikzcd}$$



Relying on the identity and associativity law as well as the axioms of finite products, according to the Equations (24)–(26), we generate a family of equivalence relations in $\mathtt{FP}(\Sigma)(X_n, X_m)$ for all $n, m \in \mathbb{N}$ that is compatible with symbolic composition and symbolic tupling. We construct the corresponding equivalence classes of fp-expressions and define composition, identity and tupling operations on these equivalence classes in the usual way by representatives. What we obtain, finally, is a finite product category $\mathsf{FP}(\Sigma)$ with equivalence classes of symbolic tuples as morphisms.

Adapting the recipe from Definition 3.14, we can obviously translate every Σ-term $t \in T_\Sigma(X_n)$ into an fp-expression $pe(t)$.

Definition 5.1 (Translation of Terms into Finite Product Expressions). *For any set X_n, $n \in \mathbb{N}$, we inductively define for all Σ-terms $t \in T_\Sigma(X_n)$ a corresponding finite product expression $pe(t)\colon X_n \to X_1$ as follows:*

Variables: $pe(\ulcorner x_i^n \urcorner) := \ulcorner x_i^n \urcorner$, *for all $n \geq 1, 1 \leq i \leq n$.*

Constants: $pe(\ulcorner \mathtt{c}\langle\rangle \urcorner) := \ulcorner \langle\rangle ; \mathtt{c} \urcorner$, *for all $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X_n)$.*

Operations: $pe(\ulcorner \mathtt{op}\langle t_1,\dots,t_n\rangle \urcorner) := \ulcorner \langle pe(t_1),\dots,pe(t_n)\rangle ; \mathtt{op}\urcorner$, *for all* $\ulcorner \mathtt{op}\langle t_1,\dots,t_n\rangle\urcorner \in T_\Sigma(X_n)$, $n \geq 1$.

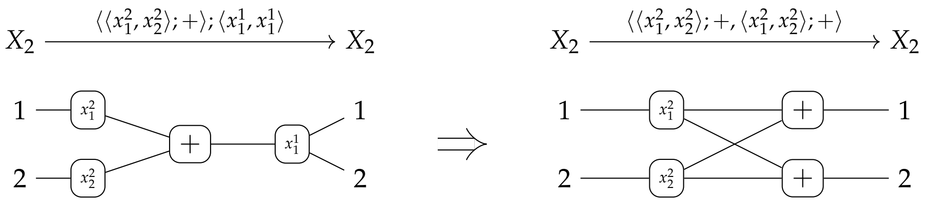
We call all the fp-expressions $pe(t)$, obtained by Definition 5.1, *fp-expressions in normal form*. Correspondingly, all the fp-expressions $\ulcorner\langle pe(t_1),\dots,pe(t_n)\rangle\urcorner$ are called *symbolic tuples in normal form*. We use the term "normal form" since they are in *normal form* with respect to a rewrite system [14] consisting of the rewriting rules given by the Equations (24)–(26) read from the left to the right.

Example 5.3 (Finite Product Expressions-Normal Forms). *From the five fp-expressions in Example 5.2, the two expressions* $\ulcorner\langle x_1^2,x_2^2\rangle;+\urcorner$ *and* $\ulcorner\langle x_1^1,x_1^1\rangle;+\urcorner$ *are fp-expressions in normal form, while* $\ulcorner\langle\langle x_1^2,x_2^2\rangle;+\rangle\urcorner$ *is a symbolic tuple in normal form.*

None of the rules (24)– (26) can be applied to $\ulcorner+\urcorner$*! However, applying these rules, we can transfer the fp-expression* $\ulcorner\langle\langle x_1^2,x_2^2\rangle;+\rangle;\langle x_1^1,x_1^1\rangle;*\urcorner : X_2 \to X_1$*, which is the symbolic composition of a symbolic tuple in normal form with an fp-expression in normal form, into normal form:*

$$\begin{aligned}\langle\langle x_1^2,x_2^2\rangle;+\rangle;\langle x_1^1,x_1^1\rangle;* \quad &\Rightarrow \quad \langle\,\langle\langle x_1^2,x_2^2\rangle;+\rangle;x_1^1\,,\,\langle\langle x_1^2,x_2^2\rangle;+\rangle;x_1^1\,\rangle\,;* &(26)\\ &\Rightarrow \quad \langle\,\langle x_1^2,x_2^2\rangle;+\,,\,\langle x_1^2,x_2^2\rangle;+\,\rangle\,;* &(25)\end{aligned}$$

The picture below shows the result of this transformation into normal form for the relevant sub-expression $\langle\langle x_1^2,x_2^2\rangle;+\rangle;\langle x_1^1,x_1^1\rangle$.


$X_2 \xrightarrow{\langle\langle x_1^2,x_2^2\rangle;+\rangle;\langle x_1^1,x_1^1\rangle} X_2$
1
2
x_1^2
x_2^2
$+$
x_1^1
1
2
$\Longrightarrow$
$X_2 \xrightarrow{\langle\langle x_1^2,x_2^2\rangle;+,\langle x_1^2,x_2^2\rangle;+\rangle} X_2$
1
2
x_1^2
x_2^2
$+$
$+$
1
2

The general effect of normalization is that all "value copying" is moved to the beginning, while we have to "clone computations units" to get rid of "value copying" happening elsewhere.

The crucial observation is that every equivalence class of symbolic tuples, constituting a morphism in $\mathsf{FP}(\Sigma)$, contains exactly one symbolic tuple in normal form! Based on this observation, it can be shown that the category $\mathsf{FP}(\Sigma)$ is isomorphic to the syntactic Lawvere category $\mathsf{L}(\Sigma)$.

5.1.3. Substitutions Revisited

Since every equivalence class of symbolic tuples in $\mathsf{FP}(\Sigma)$ contains exactly one symbolic tuple in normal form, we can define a corresponding *representation category* $\mathsf{FP}_{\mathsf{nf}}(\Sigma)$ with morphisms all symbolic tuples in normal form.

However, the symbolic composition of a symbolic tuple in normal form with an fp-expression in normal form or a symbolic tuple in normal form does not, in general, result in an fp-expression in normal form or a symbolic tuple in normal form, respectively. We have to normalize this composite fp-expression to define composition in $\mathsf{FP}_{\mathsf{nf}}(\Sigma)$.

The rules, given by the Equations (24)–(26), are not sufficient to transform any fp-expression into its normal form. They are, however, sufficient to compute the normal form of all symbolic compositions of a symbolic tuple in normal form with an fp-expression in normal form and thus, due to Equation (26), all symbolic compositions of symbolic tuples in normal form! We described an example of a normalization of a symbolic composition of a symbolic tuple in normal form with an fp-expression in normal form in Example 5.3.

We now have a chain of isomorphisms between categories $\mathsf{L}(\Sigma) \cong \mathsf{FP}(\Sigma) \cong \mathsf{FP}_{\mathsf{nf}}(\Sigma)$. The morphisms in $\mathsf{L}(\Sigma)$ are tuples of terms representing *substitution declarations*, while composition is nothing but *substitution application*. The tuples of terms in $\mathsf{L}(\Sigma)$ are transformed into symbolic tuples in normal form in $\mathsf{FP}_{\mathsf{nf}}(\Sigma)$, while composition in $\mathsf{FP}_{\mathsf{nf}}(\Sigma)$ is given by *symbolic composition* plus *normalization*. So, in light of substitution calculi, as discussed in Section 3.4.1, we obtain the following correspondence of concepts: "substitution declaration" ≅ "symbolic tuple in normal form". Moreover, Lawvere's original slogan "composition is substitution" turns into the slogan

substitution application ≅ symbolic composition plus normalization.

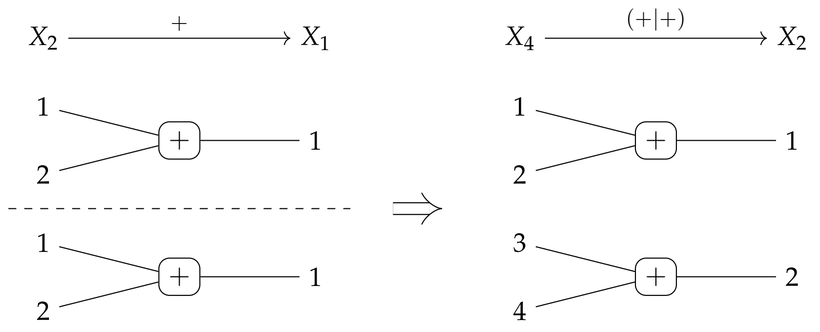
This perception may open a path to develop, in the future, an appropriate substitution calculus for derived graph operations.

5.2. Analysis of Finite Product Expressions

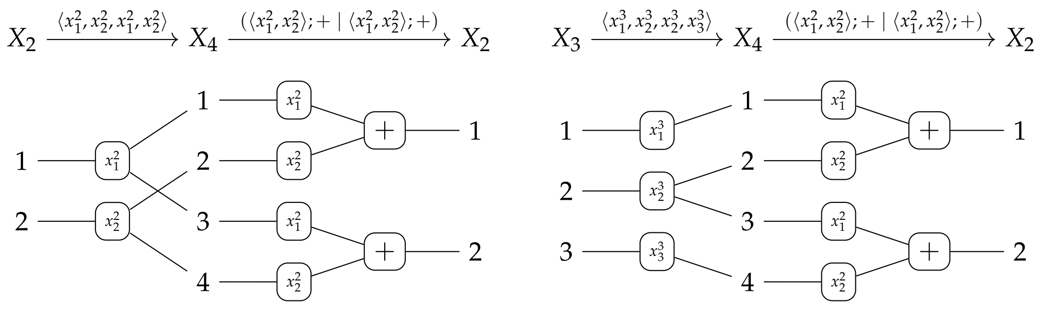
After transforming the syntactic Lawvere category $\mathsf{L}(\Sigma)$ into the isomorphic categories $\mathsf{FP}(\Sigma)$ and $\mathsf{FP}_{\mathsf{nf}}(\Sigma)$, we are now able to attack the problems, pointed at in Example 5.1, by analyzing in more detail finite product expressions.

5.2.1. Finite Products vs. Tensor Products

It is well-known that finite products also give us *tensor products* at hand [21]. We will use the term *parallel composition of morphisms* instead of *tensor product of morphisms*, and we will use the bar symbol "$|$" instead of "$\otimes$" to denote parallel composition of morphisms. The picture below visualizes the parallel composition of the fp-expression $\ulcorner + \urcorner : X_2 \to X_1$ with itself.



Conversely, *tensor products* together with *copying*, allow us to define finite products [21]. In our present setting, *copying* is represented by symbolic tuples of symbolic projections, and thus each non-empty symbolic tuple $\ulcorner\langle ex_1, \ldots, ex_n\rangle\urcorner$ can equivalently be described by a symbolic composition $\ulcorner\langle copy\rangle; (ex'_1 | \ldots | ex'_n)\urcorner$ of a symbolic tuple $\ulcorner\langle copy\rangle\urcorner$ of symbolic projections with a parallel composition $\ulcorner(ex'_1 | \ldots | ex'_n)\urcorner$ of expressions. The picture below shows the result of this transformation for the fp-expression $\langle\langle x_1^2, x_2^2\rangle; +, \langle x_1^2, x_2^2\rangle; +\rangle$, discussed in Example 5.3. To exemplify that we do not always have $ex'_i = ex_i$, we also show the result for the variant $\langle\langle x_1^3, x_2^3\rangle; +, \langle x_2^3, x_3^3\rangle; +\rangle$.


$X_2 \xrightarrow{\langle x_1^2, x_2^2, x_1^2, x_2^2\rangle} X_4 \xrightarrow{(\langle x_1^2, x_2^2\rangle; + \mid \langle x_1^2, x_2^2\rangle; +)} X_2$
$X_3 \xrightarrow{\langle x_1^3, x_2^3, x_2^3, x_3^3\rangle} X_4 \xrightarrow{(\langle x_1^2, x_2^2\rangle; + \mid \langle x_1^2, x_2^2\rangle; +)} X_2$

In conclusion, in the case of traditional operations, *tupling* can equivalently be replaced by *parallel composition* plus *copying*. There are no problems concerning parallel composition of graph operations, and thus the problem with tupling can finally be shown to be a problem with copying. We have to eventually replace *copying* by another mechanism that does not cause problems!

5.2.2. Copying vs. Soldering

How can we explain, in terms of computation diagrams, the effect of pre-composing an expression $ex : X_n \to X_m$ with a symbolic tuple $\ulcorner\langle copy\rangle\urcorner : X_k \to X_n$ of symbolic projections? We construct out of a computation diagram with n input ports and m output ports a new computation diagram with k input ports and the same m output ports.

To explain this construction, we have to leave the pure world of expressions and remember that a symbolic tuple $\ulcorner\langle copy\rangle\urcorner : X_k \to X_n$ of symbolic projections encodes a map $\boldsymbol{c} : X_n \to X_k$ between ports. If $\boldsymbol{c}$ is not surjective, the construction adds each element in $X_k \setminus \boldsymbol{c}(X_n)$ as a "dummy input port". In addition, each original input port $x \in X_n$ is soldered with all other input ports $x' \in X_n$ with $\boldsymbol{c}(x') = \boldsymbol{c}(x)$ to a single input port in X_k. Relying on our conventions in Section 2 concerning the notation of maps, we will use a new type of expression $[\boldsymbol{c}](ex)$ to denote the new operation from X_k to X_m defined by the newly constructed computation diagram and call it the *instance of* $\ulcorner ex\urcorner$ *with respect to* $\boldsymbol{c}$. The left picture in Figure 6 visualizes the construction of $[1,2,1,2](+|+)$.

In the case of *computation units* and non-injective, surjective maps $\boldsymbol{c} : X_n \to X_k$, we could even interpret the construction of $[\boldsymbol{c}](ex)$ as the construction of a new computation unit computing, for example, the square of a number (see the right picture in Figure 6).

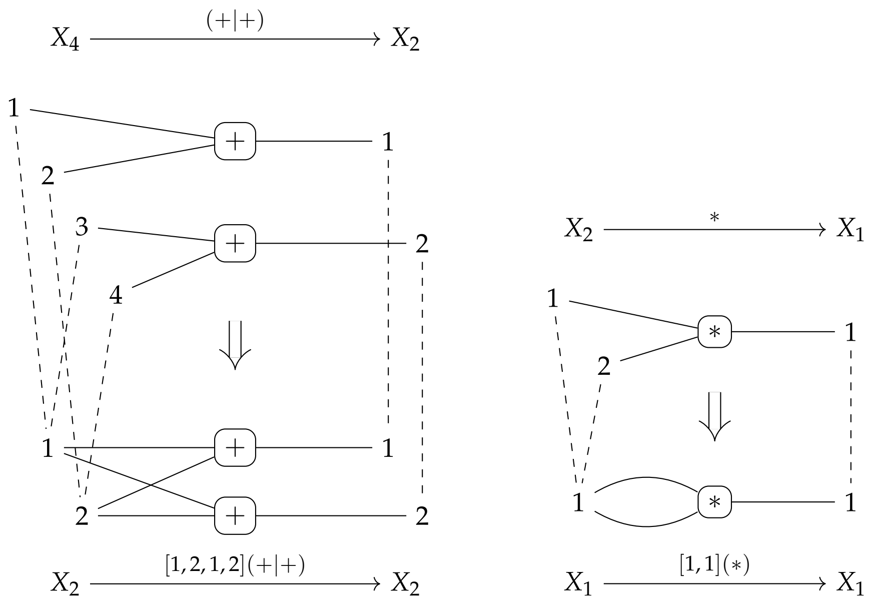

$X_4 \xrightarrow{(+|+)} X_2$
1
2
3
4
+
+
1
2
⇓
1
2
+
+
1
2
$X_2 \xrightarrow{[1,2,1,2](+|+)} X_2$
$X_2 \xrightarrow{*} X_1$
1
2
*
1
⇓
1
*
1
$X_1 \xrightarrow{[1,1](*)} X_1$

Figure 6. Instances of computation diagrams.

In the case of traditional operations, the soldering of input ports has no effect on output ports since the boundaries are empty! In the case of graph operations, however, soldering of input ports may cause soldering of items in the boundary and thus, potentially, also of output ports. This is exactly the mechanism we have been looking for to solve the problems with tupling exemplified in Example 5.1, as we will demonstrate in Section 5.3.

5.3. Three Mechanisms to Construct New Graph Operations

Our analysis in the last subsection suggests that we should try to define "derived graph operations" by means of three basic constructions on graph operations—parallel composition, instantiation, and sequential composition.

5.3.1. Parallel Composition

Given a graph $\mathtt{G}$ and a family $\omega_i\colon \mathtt{G}^{\mathtt{I}_i} \to \mathtt{G}^{\mathtt{O}_i}, 1 \leq i \leq k, k \geq 2$ of graph operations with arity spans $ar_i = \mathtt{I}_i \xleftarrow{l_i} \mathtt{B}_i \xrightarrow{r_i} \mathtt{O}_i$, we can construct a new graph operation

$$\omega := \omega_1 + \ldots + \omega_k\colon \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}} \quad \text{with arity } ar = ar_1 + \ldots + ar_k = \mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O} \ , \tag{27}$$

called the *parallel composition* of $\omega_1, \ldots, \omega_k$, where the arity graphs are given by sums of graphs $\mathtt{I} = \mathtt{I}_1 + \ldots + \mathtt{I}_k$, $\mathtt{B} = \mathtt{B}_1 + \ldots + \mathtt{B}_k$, $\mathtt{O} = \mathtt{O}_1 + \ldots + \mathtt{O}_k$ and the inclusion graph homomorphims are sums of graph homomorphisms $l = l_1 + \ldots + l_k$, $r = r_1 + \ldots + r_k$.

The sum $\mathtt{I} = \mathtt{I}_1 + \ldots + \mathtt{I}_k$ comes along with a family $\kappa_i^{\mathtt{I}} : \mathtt{I}_i \to \mathtt{I}$, $1 \leq i \leq k$ of injections and for any $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}}$ the uniqueness of mediating morphisms entails the equation

$$\boldsymbol{b} = [\kappa_1^{\mathtt{I}};\boldsymbol{b}, \ldots, \kappa_k^{\mathtt{I}};\boldsymbol{b}] : \mathtt{I} \to \mathtt{G}\,. \tag{28}$$

By applying the given operations $\omega_i\colon \mathtt{G}^{\mathtt{I}_i} \to \mathtt{G}^{\mathtt{O}_i}$, $1 \leq i \leq k$, we obtain a family $\omega_i(\kappa_i^{\mathtt{I}};\boldsymbol{b}) : \mathtt{O}_i \to \mathtt{G}$ of graph homomorphisms satisfying the commutativity requirement for graph operations in Definition 4.2:

$$l_i;\kappa_i^{\mathtt{I}};\boldsymbol{b} = r_i;\omega_i(\kappa_i^{\mathtt{I}};\boldsymbol{b}) \quad \text{for all } 1 \leq i \leq k. \tag{29}$$

We define the result of applying $\omega = \omega_1 + \ldots + \omega_k$ to an input $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}}$ as the unique cotuple of the single results

$$\omega(\boldsymbol{b}) = (\omega_1 + \ldots + \omega_k)(\boldsymbol{b}) := [\omega_1(\kappa_1^{\mathtt{I}};\boldsymbol{b}), \ldots, \omega_k(\kappa_k^{\mathtt{I}};\boldsymbol{b})]\colon \mathtt{O} \to \mathtt{G}. \tag{30}$$

The algebraic laws of cotuples and sums ensure that the commutativity requirement for graph operations is satisfied:

$$\begin{array}{rcll}
r;\omega(\boldsymbol{b}) & = & (r_1 + \ldots + r_k);[\omega_1(\kappa_1^{\mathtt{I}};\boldsymbol{b}), \ldots, \omega_k(\kappa_k^{\mathtt{I}};\boldsymbol{b})] & (\text{def. of } r \text{ and } (30)) \\
 & = & [r_1;\omega_1(\kappa_1^{\mathtt{I}};\boldsymbol{b}), \ldots, r_k;\omega_k(\kappa_k^{\mathtt{I}};\boldsymbol{b})] & \\
 & = & [l_1;\kappa_1^{\mathtt{I}};\boldsymbol{b}, \ldots, l_k;\kappa_k^{\mathtt{I}};\boldsymbol{b}] & (29) \\
 & = & (l_1 + \ldots + l_k);[\kappa_1^{\mathtt{I}};\boldsymbol{b}, \ldots, \kappa_k^{\mathtt{I}};\boldsymbol{b}] & \\
 & = & l;\boldsymbol{b} & (\text{def. of } l \text{ and } (28))
\end{array}$$

Remark 5.2 (Parallel Composition - Index Shifting). *In the case that all the arity spans* $ar_i = \mathtt{I}_i \xleftarrow{l_i} \mathtt{B}_i \xrightarrow{r_i} \mathtt{O}_i$ *are canonical arity spans, in the sense of Remark 4.2, we can also construct the arity* $ar = ar_1 + \ldots + ar_k = \mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$ *as a canonical arity span.*

We construct the sums of sets $\mathtt{I}_V = (\mathtt{I}_1)_V + \ldots + (\mathtt{I}_k)_V$ *and* $\mathtt{I}_E = (\mathtt{I}_1)_E + \ldots + (\mathtt{I}_k)_E$ *utilizing the technique of "index shifting" we used in Section 3.7.2 to define finite products in Lawvere theories. The sum* $\mathtt{B} = \mathtt{B}_1 + \ldots + \mathtt{B}_k$ *of boundary graphs can be chosen to be a corresponding subgraph of* $\mathtt{I} = \mathtt{I}_1 + \ldots + \mathtt{I}_k$*. Finally, we can construct a sum* $\mathtt{O} = \mathtt{O}_1 + \ldots + \mathtt{O}_k$ *with* $\mathtt{O}_V = \mathtt{B}_V \cup (\mathtt{O}_V \setminus \mathtt{B}_V)$ *and* $\mathtt{O}_E = \mathtt{B}_E \cup (\mathtt{O}_E \setminus \mathtt{B}_E)$ *where both sums of sets* $\mathtt{O}_V \setminus \mathtt{B}_V := (\mathtt{O}_1)_V \setminus (\mathtt{B}_1)_V + \ldots + (\mathtt{O}_k)_V \setminus (\mathtt{B}_k)_V$ *and* $\mathtt{O}_E \setminus \mathtt{B}_E := (\mathtt{O}_1)_E \setminus (\mathtt{B}_1)_E + \ldots + (\mathtt{O}_k)_E \setminus (\mathtt{B}_k)_E$ *are again constructed by means of "index shifting".*

The technique of "index shifting" allows us to define sums of canonical arities in such a way that the formation of these sums becomes associative. This means, especially, that there is no need to work with "nested parallel compositions".

Example 5.4 (Parallel Composition). *We consider a* Γ_{cat}*-algebra* $\mathcal{C} = (gr(\mathsf{C}), OP^{\mathcal{C}})$ *given by a category* C *as described in Example 4.2. The upper part of Figure 14 is the same as the upper part of Figure 7 and shows the arity* $ar(comp) + ar(comp)$ *of the parallel composition* $\mathtt{comp}^{\mathcal{C}} + \mathtt{comp}^{\mathcal{C}}$ *of the composition operation in* $\mathcal{C}$ *with itself.*

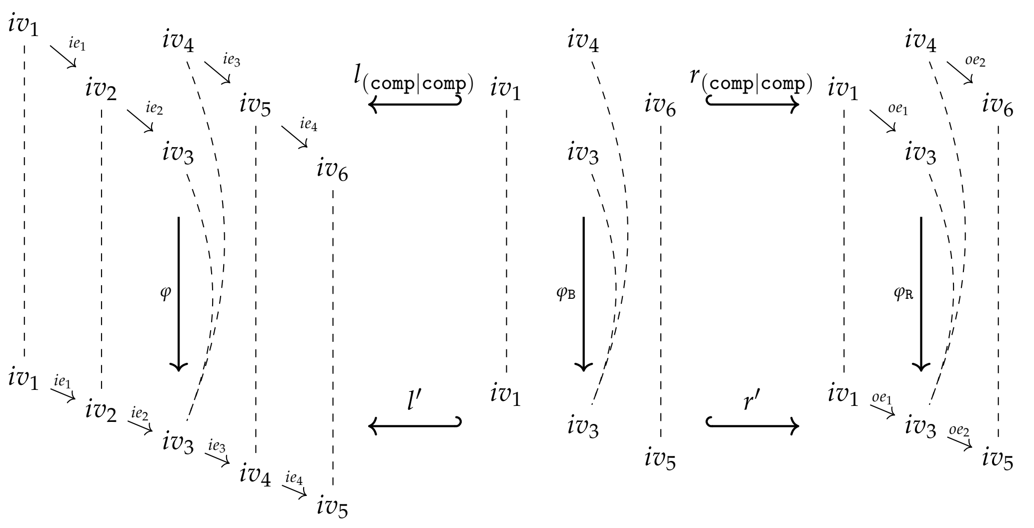


Figure 7. Parallel application of two composition operations on successive pairs of arrows.

Analogously, the upper part of Figure 8 shows the arity of the parallel composition $\texttt{comp}^{\mathcal{C}} + \pi^{\mathtt{I}}_{\mathtt{I}}$ *of the composition operation in* $\mathcal{C}$ *with the built-in projection (identity map)* $\pi^{\mathtt{I}}_{\mathtt{I}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{I}}$*, as defined in Remark 4.3, where* $\mathtt{I}$ *is the canonical input arity graph* $iv_1 \xrightarrow{ie_1} iv_2$ *(the notations for "graph operation expressions", like* $(\texttt{comp}|\texttt{comp})$ *and* $(\texttt{comp}|(\mathtt{I},\mathtt{I}))$*, will be defined in Section 5.4).*

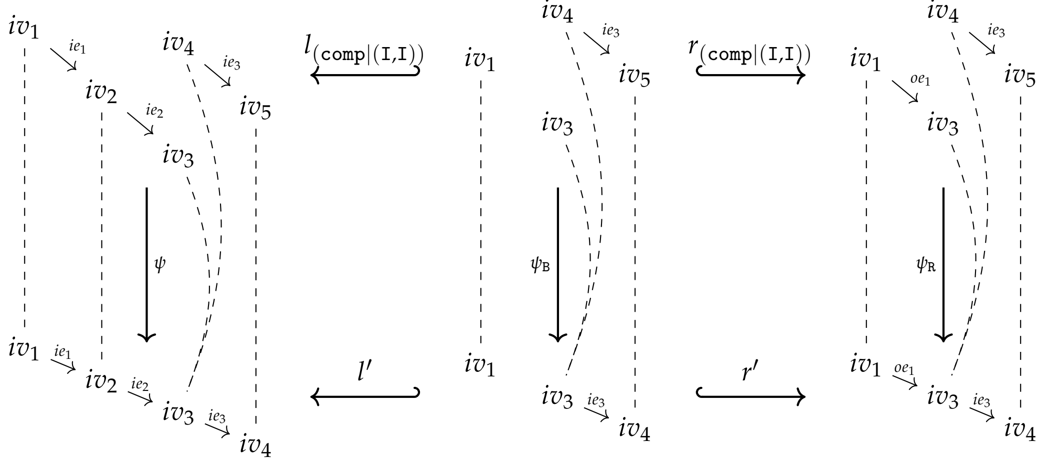


Figure 8. Parallel application of a composition operation and an identity map on arrows.

5.3.2. Instantiation

Given a graph $\mathtt{G}$ and a graph operation $\omega\colon\ \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$ with arity $ar = \ \mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$, we can construct for any finite graph $\mathtt{I}'$ and any graph homomorphism $\varphi : \mathtt{I} \to \mathtt{I}'$ a graph operation

$$\omega/\varphi :=\colon\ \mathtt{G}^{\mathtt{I}'} \to \mathtt{G}^{\mathtt{O}'} \quad \text{with arity } ar/\varphi = \ \mathtt{I}' \xleftarrow{l'} \mathtt{B}' \xrightarrow{r'} \mathtt{O}' \ , \tag{31}$$

called the *instance of* ω *with respect to* φ, where $\mathtt{B}' := \varphi(\mathtt{B}) \sqsubseteq \mathtt{I}'$ and $\mathtt{O}'$ is constructed as a pushout of the span $\mathtt{B}' \xleftarrow{\varphi_{\mathtt{B}}} \mathtt{B} \xrightarrow{r} \mathtt{O}$ with $\mathtt{B}' \sqsubseteq \mathtt{O}'$ such that $\mathtt{I}'$ and $\mathtt{O}' \setminus \mathtt{B}'$ are disjoint as depicted in the picture below. For any input $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}'}$, we can apply the given graph operation $\omega\colon\ \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$ to $\varphi;\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}}$ and obtain a result $\omega(\varphi;\boldsymbol{b}) \in \mathtt{G}^{\mathtt{O}}$ such that $l;\varphi;\boldsymbol{b} = r;\omega(\varphi;\boldsymbol{b})$. Since

$l;\varphi = \varphi_{\mathtt{B}};l'$ by construction, we obtain $\varphi_{\mathtt{B}};l';\boldsymbol{b} = r;\omega(\varphi;\boldsymbol{b})$ and thus the pushout property of the right square entails the existence of a unique $k_{\boldsymbol{b}} : \mathtt{O}' \to \mathtt{G}$ such that $l';\boldsymbol{b} = r';k_{\boldsymbol{b}}$ and $\varphi_{\mathtt{O}};k_{\boldsymbol{b}} = \omega(\varphi;\boldsymbol{b})$.

$$\begin{tikzcd}
\mathtt{I} \arrow[d, "\varphi"] \arrow[dd, bend right=60, "\varphi;\boldsymbol{b}"'] \arrow[dr, phantom, "\circlearrowleft"] & \mathtt{B} \arrow[l, hook', "l"'] \arrow[r, hook, "r"] \arrow[d, "\varphi_{\mathtt{B}}"] \arrow[dr, phantom, "po"] & \mathtt{O} \arrow[d, "\varphi_{\mathtt{O}}"] \arrow[dl, bend left=60, "\omega(\varphi;\boldsymbol{b})", to path={-- (\tikztotarget)}, phantom] \\
\mathtt{I}' \arrow[dr, "\boldsymbol{b}"] & \mathtt{B}' \arrow[l, hook', "l'"'] \arrow[r, hook, "r'"] \arrow[d, phantom, "\circlearrowleft"] & \mathtt{O}' \arrow[dl, dashed, "k_{\boldsymbol{b}}"] \\
& \mathtt{G} &
\end{tikzcd}$$

It is probably worth mentioning that it is sufficient to require that $\varphi_{\mathtt{O}};k_{\boldsymbol{b}} = \omega(\varphi;\boldsymbol{b})$. $\varphi_{\mathtt{B}} : \mathtt{B} \to \mathtt{B}'$ is an epimorphism by construction such that $\varphi_{\mathtt{O}} : \mathtt{O} \to \mathtt{O}'$ also becomes an epimorphism since pushouts preserve epimorphisms. Therefore, $k_{\boldsymbol{b}} : \mathtt{O}' \to \mathtt{G}$ is uniquely determined by the requirement $\varphi_{\mathtt{O}};k_{\boldsymbol{b}} = \omega(\varphi;\boldsymbol{b})$. Moreover, this requirement also implies $l';\boldsymbol{b} = r';k_{\boldsymbol{b}}$ since $\varphi_{\mathtt{B}} : \mathtt{B} \to \mathtt{B}'$ is an epimorphism. In other words, $\omega(\varphi;\boldsymbol{b})$ factors uniquely through $\varphi_{\mathtt{O}} : \mathtt{O} \to \mathtt{O}'$ and thus the term "instance" is indeed appropriate.

For any input $\boldsymbol{b} \in \mathtt{G}^{\mathtt{I}'}$,. we define $(\omega/\varphi)(\boldsymbol{b}) := k_{\boldsymbol{b}}$. This ensures, especially, the required commutativity $l';\boldsymbol{b} = r';(\omega/\varphi)(\boldsymbol{b})$.

Remark 5.3 (Instantiation—Canonical Arities)**.** *For a canonical arity span* $\mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$ *and a canonical input arity graph* $\mathtt{I}'$*, we have that* $\mathtt{I}'$ *and* $\mathtt{O} \setminus \mathtt{B}$ *are disjoint. In such a way, we can simply define* $\mathtt{O}'_V = \mathtt{B}'_V \cup (\mathtt{O}_V \setminus \mathtt{B}_V)$ *and* $\mathtt{O}'_E = \mathtt{B}'_E \cup (\mathtt{O}_E \setminus \mathtt{B}_E)$ *such that* $\mathtt{I}' \xleftarrow{l'} \mathtt{B}' \xrightarrow{r'} \mathtt{O}'$ *becomes a canonical arity span as well.*

Example 5.5 (Instances of the Composition Operation)**.** *There are four different instances of the composition operation* $\mathtt{comp}^{\mathcal{C}}$ *in a* Γ_{cat}*-algebra* $\mathcal{C} = (gr(\mathtt{C}), OP^{\mathcal{C}})$ *as described in Example 4.2.*

The lower part of Figure 9 visualizes the arity $ar(comp)/\beta$ *of the instance* $\mathtt{comp}^{\mathcal{C}}/\beta$ *of* $\mathtt{comp}^{\mathcal{C}}$*, giving us the composition of two loops at hand. The arity* $ar(comp)/\gamma$ *of the most specialized instance* $\mathtt{comp}^{\mathcal{C}}/\gamma$ *of* $\mathtt{comp}^{\mathcal{C}}$*, namely the composition of a loop with itself, is shown in the lower part of Figure 10.*

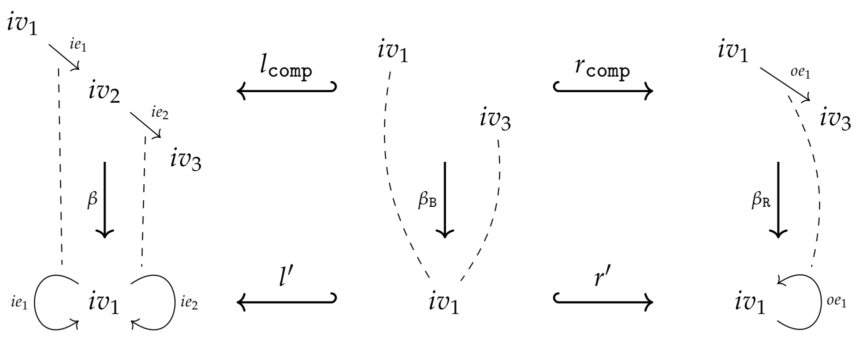


Figure 9. Composition of two loops.

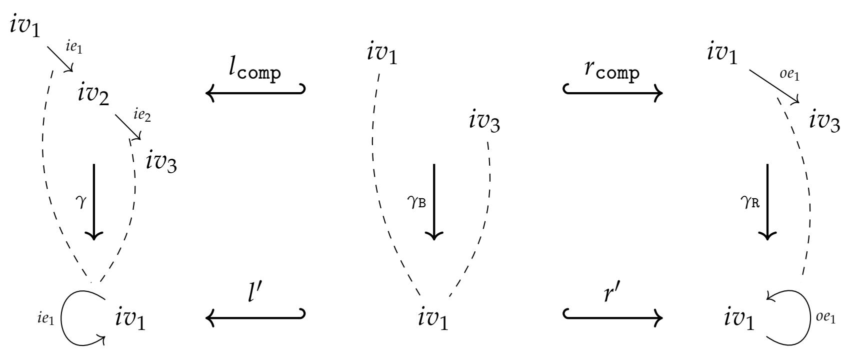


Figure 10. Composition of a loop with itself.

Example 5.6 (Tupling versus Soldering)**.** *The lower part of Figure 7 visualizes the arity of the instance* $(\mathtt{comp}^{\mathcal{C}} + \mathtt{comp}^{\mathcal{C}}) / \varphi$ *of the parallel composition* $\mathtt{comp}^{\mathcal{C}} + \mathtt{comp}^{\mathcal{C}}$ *in Example 5.4. It shows that instantiation indeed provides the "soldering effect" we need to solve the problems with "tupling" as exemplified in Example 5.1! The output arity of the graph operation* $(\mathtt{comp}^{\mathcal{C}} + \mathtt{comp}^{\mathcal{C}}) / \varphi$ *consists of two successive arrows; thus, we can indeed compose it with* $\mathtt{comp}^{\mathcal{C}}$ *(see Example 5.7).*

Analogously, the lower part of Figure 8 visualizes the arity of the instance $(\mathtt{comp}^{\mathcal{C}} + \pi^{\mathtt{I}}_{\mathtt{I}}) / \psi$ *of the parallel composition* $\mathtt{comp}^{\mathcal{C}} + \pi^{\mathtt{I}}_{\mathtt{I}}$ *in Example 5.4. This example shows that we also need "soldering" to describe the composition of three arrows by means of "derived graph operations".*

Remark 5.4 (Transfer of Items)**.** *Note that* φ *is not required to be surjective! The items in* $\mathtt{I}' \setminus \varphi(\mathtt{I})$ *have no influence on the output produced by* ω / φ *and are ignored. Specifically, they do not appear in* $\mathtt{B}' = \varphi(\mathtt{B})$ *and thus not in* $\mathtt{O}'$ *either.*

Any transfer of items from the input to the output has to be performed explicitly! If we need to transfer, in addition, also items from $\mathtt{I}' \setminus \varphi(\mathtt{B})$ *to the output, we have first to construct a parallel composition of* ω *with appropriate built-in projections before we define a corresponding extended* φ' *that also comprises the items in* $\mathtt{I}' \setminus \varphi(\mathtt{B})$ *we wish to transfer. An example for such a "need of transfer" is the successive composition of three arrows (compare Figure 8).*

Corollary 5.1 (Instantiation)**.** *Let a graph* $\mathtt{G}$ *and a graph operation* $\omega \colon \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$ *with a canonical arity span* $\mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$ *be given. For any canonical input arity graphs* $\mathtt{I}', \mathtt{I}''$ *and any graph homomorphisms* $\varphi : \mathtt{I} \to \mathtt{I}'$, $\psi : \mathtt{I}' \to \mathtt{I}''$*, we have*

$$(\omega / \varphi) / \psi = \omega / (\varphi; \psi).$$

Proof. This follows immediately from the fact that $(\varphi; \psi)(\mathtt{I}) = \psi(\varphi(\mathtt{I}))$ and the fact that the composition of two pushouts is also a pushout. The choice of canonical arities ensures, especially, that $\mathtt{O}_V \setminus \mathtt{B}_V = \mathtt{O}'_V \setminus \mathtt{B}'_V = \mathtt{O}''_V \setminus \mathtt{B}''_V$ and $\mathtt{O}_E \setminus \mathtt{B}_E = \mathtt{O}'_E \setminus \mathtt{B}'_E = \mathtt{O}''_E \setminus \mathtt{B}''_E$, by construction. □

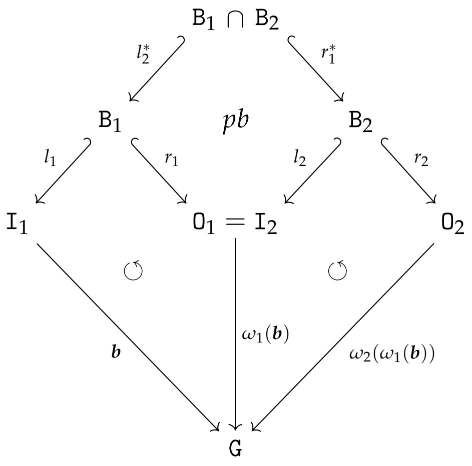
Hypothesis 5.1 (Parallel Composition and Instantiation)**.** *It should not be a problem to prove a more general result about the interplay of parallel composition and instantiation.*

We consider a graph $\mathtt{G}$ *and a family* $\omega_i \colon \mathtt{G}^{\mathtt{I}_i} \to \mathtt{G}^{\mathtt{O}_i}$, $1 \leq i \leq k$, $k \geq 2$ *of graph operations with canonical arity spans* $\mathtt{I}_i \xleftarrow{l_i} \mathtt{B}_i \xrightarrow{r_i} \mathtt{O}_i$ *together with a family* $\varphi_i : \mathtt{I}_i \to \mathtt{I}'_i$, $1 \leq i \leq k$ *of homomorphisms where all the* $\mathtt{I}'_i$ *values are canonical input arity graphs. For any graph homomorphism* $\varphi : \mathtt{I}'_1 + \ldots + \mathtt{I}'_k \to \mathtt{I}''$ *with* $\mathtt{I}''$ *a canonical input arity graph, we have*

$$(\omega_1 / \varphi_1 + \ldots + \omega_k / \varphi_k) / \varphi = (\omega_1 + \ldots + \omega_k) / (\varphi_1 + \ldots + \varphi_k); \varphi.$$

5.3.3. Sequential Composition

At first glance, it should not be a problem to sequentially compose two graph operations $\omega_1\colon\ \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{O}_1}$ with arity $ar_1 = \ \mathtt{I}_1 \xhookleftarrow{l_1} \mathtt{B}_1 \xhookrightarrow{r_1} \mathtt{O}_1$ and $\omega_2\colon\ \mathtt{G}^{\mathtt{I}_2} \to \mathtt{G}^{\mathtt{O}_1}$ with arity $ar_2 = \ \mathtt{I}_2 \xhookleftarrow{l_2} \mathtt{B}_2 \xhookrightarrow{r_2} \mathtt{O}_2$. We simply require $\mathtt{O}_1 = \mathtt{I}_2$ and define, as usual, composition through successive application, $(\omega_1;\omega_2)(\boldsymbol{b}) := \omega_2(\omega_1(\boldsymbol{b}))$ for all $\boldsymbol{b} \in \mathtt{G}_{\mathtt{I}_1}$, while the arity of the sequential composition is given by a standard pullback based composition of spans (see the picture below).

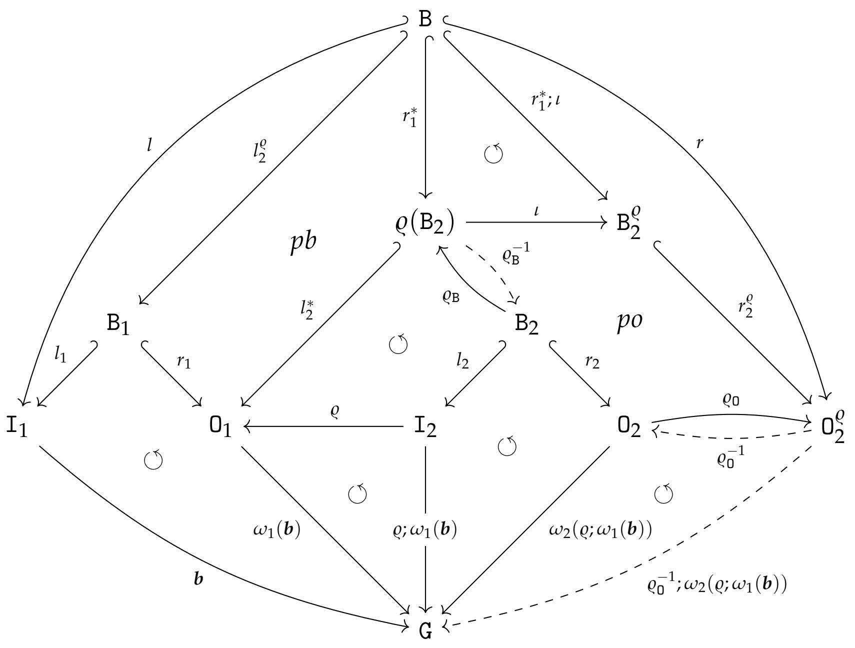

$$\begin{tikzcd}
 & & \mathtt{B}_1 \cap \mathtt{B}_2 \arrow[dl, hook', "l_2^*"'] \arrow[dr, hook, "r_1^*"] & & \\
 & \mathtt{B}_1 \arrow[dl, hook', "l_1"'] \arrow[dr, hook, "r_1"] & pb & \mathtt{B}_2 \arrow[dl, hook', "l_2"'] \arrow[dr, hook, "r_2"] & \\
\mathtt{I}_1 \arrow[ddrr, "\boldsymbol{b}"'] & & \mathtt{O}_1 = \mathtt{I}_2 \arrow[dd, "\omega_1(\boldsymbol{b})"] & & \mathtt{O}_2 \arrow[ddll, "\omega_2(\omega_1(\boldsymbol{b}))"] \\
 & \circlearrowleft & & \circlearrowleft & \\
 & & \mathtt{G} & &
\end{tikzcd}$$

There are, however, at least two problems with this naive proposal:

1. For canonical arity spans ar_1 and ar_2 , we can have an equality $\mathtt{O}_1 = \mathtt{I}_2$ only if ω_1 is a built-in projection.
2. There are two kinds of output items produced via the sequential composition of two graph operations. First, the output items produced by ω_2. Second, the output items produced by ω_1 and implicitly transferred by ω_2, i.e., the items in $\mathtt{B}_2 \setminus (\mathtt{B}_1 \cap \mathtt{B}_2) = \mathtt{B}_2 \setminus \mathtt{B}_1$. In other words, the resulting arity will not satisfy the disjointness condition in Definition 4.1 if $\mathtt{B}_2 \setminus \mathtt{B}_1$ is non-empty!

Since we intend to define graph operation expressions with canonical arity spans only, we consider canonical arity spans ar_1 and ar_2. To solve the first problem, we introduce *sequential composition via arity renaming*, i.e., instead of $\mathtt{O}_1 = \mathtt{I}_2$, we assume a graph isomorphism $\varrho : \mathtt{I}_2 \to \mathtt{O}_1$ and define a graph operation $\omega_1;^{\varrho}\omega_2 : \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{O}_2^{\varrho}}$ with arity $ar = \ \mathtt{I}_1 \xhookleftarrow{l} \mathtt{B} \xhookrightarrow{r} \mathtt{O}_2^{\varrho}$ where $\mathtt{O}_2^{\varrho}$ is isomorphic to $\mathtt{O}_2$ and constructed by means of ϱ in such a way that ar becomes a canonical arity span. This also solves the second problem.

We now describe the rather involved step-wise construction of the canonical arity of the graph operation $\omega_1;^{\varrho}\omega_2 : \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{O}_2^{\varrho}}$ obtained by the sequential composition of two given graph operations $\omega_1\colon\ \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{O}_1}$ and $\omega_2\colon\ \mathtt{G}^{\mathtt{I}_2} \to \mathtt{G}^{\mathtt{O}_1}$ via a graph isomorphism $\varrho : \mathtt{I}_2 \to \mathtt{O}_1$. The reader can follow the construction in Figure 11.



$$\begin{tikzcd}
 & & & \mathtt{B} \arrow[dddlll, hook, bend right, "l"'] \arrow[ddll, hook, "l_2^{\varrho}"'] \arrow[d, hook, "r_1^*"'] \arrow[drr, hook, "r_1^*;\iota"] \arrow[dddrrr, hook, bend left, "r"] \arrow[dd, phantom, "\circlearrowleft", shift left=6ex] & & & \\
 & & & \varrho(\mathtt{B}_2) \arrow[rr, "\iota"] \arrow[ddl, hook, "l_2^*"'] \arrow[dr, dashed, "\varrho_{\mathtt{B}}^{-1}"] \arrow[dll, phantom, "pb"] & & \mathtt{B}_2^{\varrho} \arrow[ddr, hook, "r_2^{\varrho}"] \arrow[dl, phantom, "po"] & \\
 & \mathtt{B}_1 \arrow[dl, hook, "l_1"'] \arrow[dr, hook, "r_1"] & & & \mathtt{B}_2 \arrow[ul, "\varrho_{\mathtt{B}}"] \arrow[dl, hook, "l_2"'] \arrow[dr, hook, "r_2"] & & \\
\mathtt{I}_1 \arrow[drrr, bend right, "\boldsymbol{b}"'] \arrow[rr, phantom, "\circlearrowleft", shift right=3ex] & & \mathtt{0}_1 \arrow[dr, "\omega_1(\boldsymbol{b})"'] \arrow[ur, phantom, "\circlearrowleft"] & \mathtt{I}_2 \arrow[l, "\varrho"'] \arrow[d, "\varrho;\omega_1(\boldsymbol{b})"] \arrow[rr, phantom, "\circlearrowleft", shift left=3ex] & & \mathtt{0}_2 \arrow[r, bend left, "\varrho_0"] \arrow[dll, "\omega_2(\varrho;\omega_1(\boldsymbol{b}))"] & \mathtt{0}_2^{\varrho} \arrow[l, dashed, "\varrho_0^{-1}"] \arrow[dlll, dashed, "\varrho_0^{-1};\omega_2(\varrho;\omega_1(\boldsymbol{b}))"] \\
 & & & \mathtt{G} & & &
\end{tikzcd}$$

Figure 11. Sequential composition via a graph isomorphism $\varrho : \mathtt{I}_2 \to \mathtt{0}_1$.

1. We obtain an epi-mono factorization of $l_2;\varrho : \mathtt{B}_2 \to \mathtt{I}_2$ by constructing the image of $\mathtt{B}_2$ with respect to $\varrho : \mathtt{I}_2 \to \mathtt{0}_1$. The resulting restriction $\varrho_{\mathtt{B}} : \mathtt{B}_2 \to \varrho(\mathtt{B}_2)$ of ϱ becomes an isomorphism since ϱ is an isomorphism.
2. The boundary arity of $\omega_1 ;^{\varrho} \omega_2$ can now be constructed by simple intersection (pullback): $\mathtt{B} := \mathtt{B}_1 \cap \varrho(\mathtt{B}_2)$ and $l := l_2^{\varrho}; l_1$.
3. To be able to define $\mathtt{0}_2^{\varrho}$ as an extension of $\mathtt{B}$ such that $\mathtt{0}_2^{\varrho} \setminus \mathtt{B}$ consists of canonical sets of output vertices and output edges, according to Remark 4.2, we first have to reindex the output vertices/edges in $\varrho(\mathtt{B}_2) \setminus \mathtt{B} = \varrho(\mathtt{B}_2) \setminus \mathtt{B}_1 \subseteq \mathtt{0}_1 \setminus \mathtt{B}_1$. We construct a graph $\mathtt{B}_2^{\varrho}$, isomorphic to $\varrho(\mathtt{B}_2)$ with $(\mathtt{B}_2^{\varrho})_V = \mathtt{B}_V \cup \mathtt{P}_V$ and $(\mathtt{B}_2^{\varrho})_E = \mathtt{B}_E \cup \mathtt{P}_E$, where $\mathtt{P}_V = \{ov_1, \ldots, ov_{n^{\mathtt{P}}}\}$ and $\mathtt{P}_E = \{oe_1, \ldots, oe_{m^{\mathtt{P}}}\}$ for $n^{\mathtt{P}} = |(\varrho(\mathtt{B}_2))_V \setminus \mathtt{B}_V|$ and $m^{\mathtt{P}} = |(\varrho(\mathtt{B}_2))_E \setminus \mathtt{B}_E|$.
4. We can define an isomorphism $\iota : \varrho(\mathtt{B}_2) \to \mathtt{B}_2^{\varrho}$ that is the identity on $\mathtt{B}$ such that, in addition, ι_V restricted to $(\varrho(\mathtt{B}_2))_V \setminus \mathtt{B}_V$ is an order-preserving map from $(\varrho(\mathtt{B}_2))_V \setminus \mathtt{B}_V$ to $\mathtt{P}_V$ while ι_E restricted to $(\varrho(\mathtt{B}_2))_E \setminus \mathtt{B}_E$ is an order-preserving map from $(\varrho(\mathtt{B}_2))_E \setminus \mathtt{B}_E$ to $\mathtt{P}_E$. The construction ensures that $r_1^*;\iota$ becomes an inclusion graph homomorphism $r_1^*;\iota : \mathtt{B} \to \mathtt{B}_2^{\varrho}$.
5. We construct $\mathtt{0}_2^{\varrho}$ via a pushout of the span $\mathtt{0}_2 \xleftarrow{\varrho_{\mathtt{B}}^{-1};r_2} \varrho(\mathtt{B}_2) \xrightarrow{\iota} \mathtt{B}_2^{\varrho}$ with $(\mathtt{0}_2^{\varrho})_V := \mathtt{B}_V \cup (\mathtt{P}_V + (\mathtt{0}_2)_V \setminus (\mathtt{B}_2)_V)$ where $(\mathtt{P}_V + (\mathtt{0}_2)_V \setminus (\mathtt{B}_2)_V)$ is constructed by "index shifting" and $(\mathtt{0}_2^{\varrho})_E := \mathtt{B}_E \cup (\mathtt{P}_E + (\mathtt{0}_2)_E \setminus (\mathtt{B}_2)_E)$ where $(\mathtt{P}_E + (\mathtt{0}_2)_E \setminus (\mathtt{B}_2)_E)$ is also constructed by "index shifting".
6. The construction ensures $\mathtt{B}_2^{\varrho} \sqsubseteq \mathtt{0}_2^{\varrho}$ and that $\varrho_0 : \mathtt{0}_2 \to \mathtt{0}_2^{\varrho}$ becomes an isomorphism. We set $r := r_1^*;\iota;r_2^{\varrho}$ and obtain a canonical arity span $\mathtt{I}_1 \xhookleftarrow{l} \mathtt{B} \xhookrightarrow{r} \mathtt{0}_2^{\varrho}$ as required.

After the arity of the graph operation $\omega_1 ;^{\varrho} \omega_2 : \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{0}_2^{\varrho}}$ has been constructed, we can straightforwardly define $\omega_1 ;^{\varrho} \omega_2 : \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{0}_2^{\varrho}}$ through the successive application of the given graph operations plus two intermediate arity-based isomorphic transformations:

$$(\omega_1 ;^{\varrho} \omega_2)(\boldsymbol{b}) := \varrho_0^{-1}; \omega_2(\varrho; \omega_1(\boldsymbol{b})) : \mathtt{0}_2^{\varrho} \to \mathtt{G} \quad \text{for all } \boldsymbol{b} \in \mathtt{G}^{\mathtt{I}_1}. \tag{32}$$

The commutativity condition in Definition 4.2 is trivially satisfied since the diagram in Figure 11 is commutative by construction and definition.

Example 5.7 (Composition of four and three arrows)**.** *The output arity of the graph operation* $(\texttt{comp}^{\mathcal{C}} + \texttt{comp}^{\mathcal{C}})/\varphi$ *in Example 5.6 is the graph* $\texttt{O}_1 := \; iv_1 \xrightarrow{oe_1} iv_3 \xrightarrow{oe_2} iv_5$ *(see Figure 7) while the input arity of the graph operation* $\texttt{comp}^{\mathcal{C}}$ *is the graph* $\texttt{I}_2 := \; iv_1 \xrightarrow{oe_1} iv_2 \xrightarrow{oe_2} iv_3$ *. In such a way, both graph operations can be composed via the arity renaming* $\varrho : \texttt{I}_2 \to \texttt{O}_1$ *defined by the assignments* $\{iv_1 \mapsto iv_1, iv_2 \mapsto iv_3, iv_3 \mapsto iv_5, oe_1 \mapsto oe_1, oe_2 \mapsto oe_2\}$*. The resulting graph operation* $((\texttt{comp}^{\mathcal{C}} + \texttt{comp}^{\mathcal{C}})/\varphi)\,;^{\varrho}\,\texttt{comp}^{\mathcal{C}}$ *with the canonical arity span shown in Figure 12 describes then the way of composing four successive arrows discussed in Example 5.1.*

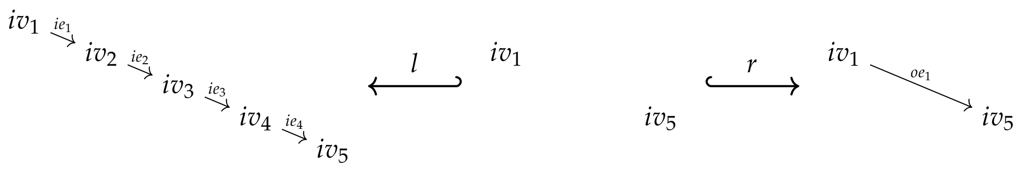

$$\begin{tikzcd}
iv_1 \arrow[dr, "ie_1"] & & & & & & & & & & & \\
& iv_2 \arrow[dr, "ie_2"] & & & & & iv_1 & & & iv_1 \arrow[ddrr, "oe_1"] & & \\
& & iv_3 \arrow[dr, "ie_3"] & & {} & {} \arrow[l, hook', "l"'] & & {} \arrow[r, hook, "r"] & {} & & & \\
& & & iv_4 \arrow[dr, "ie_4"] & & & iv_5 & & & & & iv_5 \\
& & & & iv_5 & & & & & & &
\end{tikzcd}$$

Figure 12. Arity of the composition of four arrows.

Analogously, the output arity of the graph operation $(\texttt{comp}^{\mathcal{C}} + \pi^{\texttt{I}}_{\texttt{I}})/\psi$ *in Example 5.6 is the graph* $\texttt{O}_1 \; := \; iv_1 \xrightarrow{oe_1} iv_3 \xrightarrow{ie_3} iv_4$ *(see Figure 8), while the input arity of the graph operation* $\texttt{comp}^{\mathcal{C}}$ *is the graph* $\texttt{I}_2 := \; iv_1 \xrightarrow{oe_1} iv_2 \xrightarrow{oe_2} iv_3$ *. In such a way, both graph operations can be composed via the arity renaming* $\rho : \texttt{I}_2 \to \texttt{O}_1$ *defined by the assignments* $\{iv_1 \mapsto iv_1, iv_2 \mapsto iv_3, iv_3 \mapsto iv_4, oe_1 \mapsto oe_1, oe_2 \mapsto ie_3\}$*. The resulting graph operation* $(((\texttt{comp}^{\mathcal{C}} + \pi^{\texttt{I}}_{\texttt{I}})/\psi)\,;^{\rho}\,\texttt{comp}^{\mathcal{C}}$ *with the canonical arity span shown in Figure 13 describes then the way of composing three successive arrows in a* Γ_{cat}*-algebra* $\mathcal{C} = (gr(\mathsf{C}), OP^{\mathcal{C}})$ *that corresponds to the left-hand side of the associativity law for the composition of morphisms in categories*

$$(f;g);h = f;(g;h) \quad \textit{for all } A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D \textit{ in } \mathsf{C}. \tag{33}$$

With the obvious changes, we can also construct a graph operation $(((\pi^{\texttt{I}}_{\texttt{I}} + \texttt{comp}^{\mathcal{C}})/\psi')\,;^{\rho'}\,\texttt{comp}^{\mathcal{C}}$ *with the same canonical arity span but representing the right-hand side of the associativity law.*

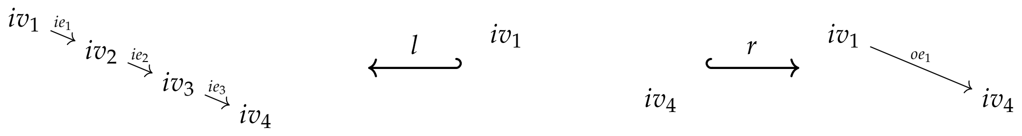
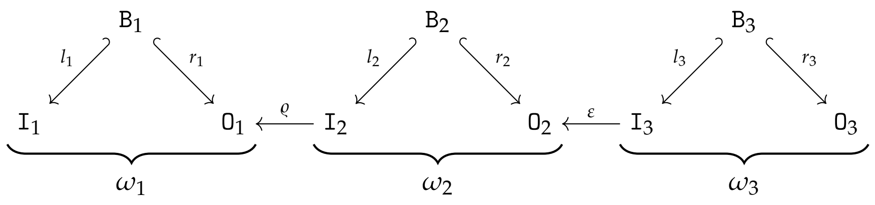

$$\begin{tikzcd}
iv_1 \arrow[dr, "ie_1"] & & & & & & & & & & & \\
& iv_2 \arrow[dr, "ie_2"] & & & & & iv_1 & & & iv_1 \arrow[ddrr, "oe_1"] & & \\
& & iv_3 \arrow[dr, "ie_3"] & & {} & {} \arrow[l, hook', "l"'] & & {} \arrow[r, hook, "r"] & {} & & & \\
& & & iv_4 & & & iv_4 & & & & & iv_4
\end{tikzcd}$$

Figure 13. Arity of the composition of three arrows.

Hypothesis 5.2 (Associativity of Sequential Composition)**.** *We consider three graph operations with two arity renamings as depicted below.*


$$\begin{tikzcd}
& \texttt{B}_1 \arrow[dl, hook', "l_1"'] \arrow[dr, hook, "r_1"] & & & \texttt{B}_2 \arrow[dl, hook', "l_2"'] \arrow[dr, hook, "r_2"] & & & \texttt{B}_3 \arrow[dl, hook', "l_3"'] \arrow[dr, hook, "r_3"] & \\
\texttt{I}_1 & & \texttt{O}_1 & \texttt{I}_2 \arrow[l, "\varrho"'] & & \texttt{O}_2 & \texttt{I}_3 \arrow[l, "\varepsilon"'] & & \texttt{O}_3 \\
& \omega_1 & & & \omega_2 & & & \omega_3 &
\end{tikzcd}$$

What kind of associativity can we gain? Constructing, first, the composition $\omega_2;^{\varepsilon}\omega_3$, we transfer the diagram of arities above into the diagram

$$\begin{tikzcd}
 & \mathtt{B}_1 \arrow[dl, hook, "l_1"'] \arrow[dr, hook, "r_1"] & & & \mathtt{B} \arrow[dl, hook, "l"'] \arrow[dr, hook, "r"] & \\
\mathtt{I}_1 & & \mathtt{O}_1 & \mathtt{I}_2 \arrow[l, "\varrho"'] & & \mathtt{O}_3^{\varepsilon} \\
 & \omega_1 & & & \omega_2;^{\varepsilon}\omega_3 &
\end{tikzcd}$$

This means, that we can sequentially compose ω_1 and $\omega_2;^{\varepsilon}\omega_3$ via $\varrho : \mathtt{I}_2 \to \mathtt{O}_1$. We obtain a graph operation $\omega_1;^{\varrho}(\omega_2;^{\varepsilon}\omega_3)$ with input arity $\mathtt{I}_1$ and output arity $(\mathtt{O}_3^{\varepsilon})^{\varrho}$.

In contrast, we cannot compose $\omega_1;^{\varrho}\omega_2$ and ω_3 via $\varepsilon : \mathtt{I}_3 \to \mathtt{O}_2$ since the output arity of $\omega_1;^{\varrho}\omega_2$ is $\mathtt{O}_2^{\varrho}$ and not $\mathtt{O}_2$ as required. Fortunately, we can bridge the gap using $\varrho_{\mathtt{O}}^{-1} : \mathtt{O}_2 \to \mathtt{O}_2^{\varrho}$ (see Figure 11 and the diagram below) and construct the sequential composition $(\omega_1;^{\varrho}\omega_2);^{\varepsilon;\varrho_{\mathtt{O}}^{-1}}\omega_3$ with input arity $\mathtt{I}_1$ and output arity $\mathtt{O}_3^{\varepsilon;\varrho_{\mathtt{O}}^{-1}}$.

$$\begin{tikzcd}
 & \mathtt{B} \arrow[dl, hook, "l_1"'] \arrow[dr, hook, "r_1"] & & & & \mathtt{B}_3 \arrow[dl, hook, "l_3"'] \arrow[dr, hook, "r_3"] & \\
\mathtt{I}_1 & & \mathtt{O}_2^{\varrho} & \mathtt{O}_2 \arrow[l, "\varrho_{\mathtt{O}}^{-1}"'] & \mathtt{I}_3 \arrow[l, "\varepsilon"'] & & \mathtt{O}_3 \\
 & \omega_1;^{\varrho}\omega_2 & & & & \omega_3 &
\end{tikzcd}$$

We are convinced that one can prove $(\mathtt{O}_3^{\varepsilon})^{\varrho} = \mathtt{O}_3^{\varepsilon;\varrho_{\mathtt{O}}^{-1}}$ and an associativity law like $\omega_1;^{\varrho}(\omega_2;^{\varepsilon}\omega_3) = (\omega_1;^{\varrho}\omega_2);^{\varepsilon;\varrho_{\mathtt{O}}^{-1}}\omega_3$, but we leave this as a topic for future research.

Remark 5.5 (Constructions and Built-in Projections)**.** *For any graph $\mathtt{G}$ the collection of all built-in projections $\pi_{\mathtt{K}}^{\mathtt{H}} : \mathtt{G}^{\mathtt{H}} \to \mathtt{G}^{\mathtt{K}}$, as described in Remark 4.3, is closed with respect to any of the three constructions—parallel composition, instantiation, or sequential composition. That is, applying any of these constructions only to built-in projections will result in a built-in projection.*

We made some effort to define the arity spans of resulting graph operations in such a way that the result of applying any of the three constructions to graph operations with canonical arity spans has a canonical arity span as well.

In such a way, the sub-collection of all built-in projections $\pi_{\mathtt{O}}^{\mathtt{I}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$ with $\mathtt{I}$ as a finite canonical input arity also becomes closed with respect to any of the three constructions. This fact may cause some redundancy when defining "graph operation expressions" and corresponding "derived graph operations". We will, however, try to avoid unnecessary redundancy.

5.4. Graph Operation Expressions and Derived Graph Operations

Now, we finally have everything at hand to define *graph operation expressions* and their semantics, i.e., the *derived graph operations* we have been looking for. We define graph operation expressions with canonical arities and with a structure as close as possible to the structure of terms. Instead of "terms" we have "single expressions", while "multi expressions" correspond to "tuples of terms".

Definition 5.2 (Graph Operation Expressions)**.** *For any graph signature $\Gamma = (OP, ar)$, we define inductively the set $GE(\Gamma)$ of all* (graph operation) Γ-expressions *with canonical arity spans.*

Projections: $\ulcorner(\mathtt{I},\mathtt{O})\urcorner \in GE(\Gamma)$ *with arity span* $ar(\ulcorner(\mathtt{I},\mathtt{O})\urcorner) = \mathtt{I} \xhookleftarrow{l} \mathtt{O} \xlongequal{r} \mathtt{O}$ *for any finite canonical input arity graph $\mathtt{I}$ and any subgraph $\mathtt{O} \sqsubseteq \mathtt{I}$.*

$\ulcorner(\mathtt{I},\mathtt{O})\urcorner$ *is called a* projection expression*.*

Constants: *$\ulcorner[\,]\mathtt{c}\urcorner \in GE(\Gamma)$ with arity $ar(\ulcorner[\,]\mathtt{c}\urcorner) = \mathtt{I} \xleftarrow{l} \mathbf{0} \xrightarrow{r} \mathtt{O}_\mathtt{c}$ for any constant symbol $\mathtt{c} \in OP$ and any finite canonical input arity graph* $\mathtt{I}$.

Moreover,

$\ulcorner([\,]\mathtt{c}\downarrow\mathtt{O})\urcorner \in GE(\Gamma)$ with arity $ar(\ulcorner([\,]\mathtt{c}\downarrow\mathtt{O})\urcorner) = \mathtt{I} \xleftarrow{l} \mathbf{0} \xrightarrow{r} \mathtt{O}$ for any non-empty subgraph $\mathtt{O} \sqsubseteq \mathtt{O}_\mathtt{c}$.

Both expressions $\ulcorner[\,]\mathtt{c}\urcorner$ and $\ulcorner([\,]\mathtt{c}\downarrow\mathtt{O})\urcorner$ are declared as single Γ-expressions.

Operations: *$\ulcorner[syn(\varphi)]\mathtt{op}\urcorner \in GE(\Gamma)$ for any operation symbol $\mathtt{op} \in OP$, any finite canonical input arity graph $\mathtt{I}'$, and any graph homomorphism $\varphi : \mathtt{I}_\mathtt{op} \to \mathtt{I}'$ where the arity $ar(\ulcorner[syn(\varphi)]\mathtt{op}\urcorner) = \mathtt{I}' \xleftarrow{l'} \mathtt{B}' \xrightarrow{r'} \mathtt{O}'$ is constructed as an instance of the arity $ar(\mathtt{op}) = \mathtt{I}_\mathtt{op} \xleftarrow{l_\mathtt{op}} \mathtt{B}_\mathtt{op} \xrightarrow{r_\mathtt{op}} \mathtt{O}_\mathtt{op}$ by means of $\mathtt{B}' := \varphi(\mathtt{B}_\mathtt{op})$ and a pushout as in (31).*

Moreover, $\ulcorner([syn(\varphi)]\mathtt{op}\downarrow\mathtt{O})\urcorner \in GE(\Gamma)$ for any non-empty subgraph $\mathtt{O} \sqsubseteq \mathtt{O}'$ such that $\mathtt{O} \setminus \mathtt{B}'$ is non-empty with arity $ar(\ulcorner([syn(\varphi)]\mathtt{op}\downarrow\mathtt{O})\urcorner) = \mathtt{I}' \xleftarrow{l'_\mathtt{O}} \mathtt{B}' \cap \mathtt{O} \xrightarrow{r'_\mathtt{O}} \mathtt{O}$.

Both expressions $\ulcorner[syn(\varphi)]\mathtt{op}\urcorner$ and $\ulcorner([syn(\varphi)]\mathtt{op}\downarrow\mathtt{O})\urcorner$ are called basic Γ-expressions *and are declared as* single Γ-expressions.

Multi expressions: *$\ulcorner(ge_1|\ldots|ge_k)\urcorner \in GE(\Gamma)$ for any family ge_i, $1 \le i \le k$, $k \ge 2$ of single Γ-expressions or projection expressions with, at least, one single Γ-expression. The arity $ar(\ulcorner(ge_1|\ldots|ge_k)\urcorner) := ar(ge_1) + \ldots + ar(ge_k) = \mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$ is constructed as described in Remark 5.2.*

Moreover, $\ulcorner[syn(\varphi)](ge_1|\ldots|ge_k)\urcorner \in GE(\Gamma)$ for any finite canonical input arity graph $\mathtt{I}' \neq \mathtt{I}$, and any graph homomorphism $\varphi : \mathtt{I} \to \mathtt{I}'$ with arity $ar(\ulcorner[syn(\varphi)](ge_1|\ldots|ge_k)\urcorner) = \mathtt{I}' \xleftarrow{l'} \mathtt{B}' \xrightarrow{r'} \mathtt{O}'$ constructed as an instance of the arity $ar(\ulcorner(ge_1|\ldots|ge_k)\urcorner) = \mathtt{I} \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}$ by means of $\mathtt{B}' := \varphi(\mathtt{B})$ and a pushout as performed in (31).

Both expressions $\ulcorner(ge_1|\ldots|ge_k)\urcorner$ and $\ulcorner[syn(\varphi)](ge_1|\ldots|ge_k)\urcorner$ are declared as multi Γ-expressions.

Symbolic composition: *$\ulcorner(ge_1;syn(\varrho);ge_2)\urcorner \in GE(\Gamma)$ for any single or multi Γ-expression ge_1 with arity $ar(ge_1) = \mathtt{I}_1 \xleftarrow{l_1} \mathtt{B}_1 \xrightarrow{r_1} \mathtt{O}_1$, any basic Γ-expression ge_2 with arity $ar(ge_2) = \mathtt{I}_2 \xleftarrow{l_2} \mathtt{B}_2 \xrightarrow{r_2} \mathtt{O}_2$, and any graph isomorphism $\varrho : \mathtt{I}_2 \to \mathtt{O}_1$.*

The arity $ar(\ulcorner(ge_1;syn(\varrho);ge_2)\urcorner) = \mathtt{I}_1 \xleftarrow{l} \mathtt{B} \xrightarrow{r} \mathtt{O}_2^\varrho$ with $\mathtt{O}_2^\varrho$ isomorphic to $\mathtt{O}_2$ is constructed as described in Section 5.3.3.

$\ulcorner(ge_1;syn(\varrho);ge_2)\urcorner$ is declared as a single Γ-expressions.

Remark 5.6 (Notational Convention: Trivial Instances)**.** *In case of trivial instances, we will just drop the corresponding substring "$[\ldots]$".*

Constants: *In the case $\mathtt{I} = \mathbf{0}$, we will just write $\ulcorner\mathtt{c}\urcorner$ instead of $\ulcorner[\,]\mathtt{c}\urcorner$ and $\ulcorner(\mathtt{c}\downarrow\mathtt{O})\urcorner$ instead of $\ulcorner([\,]\mathtt{c}\downarrow\mathtt{O})\urcorner$.*

Operations: *In the case $\varphi = id_{\mathtt{I}_\mathtt{op}}$, we will just write $\ulcorner\mathtt{op}\urcorner$ instead of $\ulcorner[syn(\varphi)]\mathtt{op}\urcorner$ and $\ulcorner(\mathtt{op}\downarrow\mathtt{O})\urcorner$ instead of $\ulcorner([syn(\varphi)]\mathtt{op}\downarrow\mathtt{O})\urcorner$.*

Since we utilize sequential composition via arity renaming, we could even require φ to be a non-isomorphism in the cases ***Operations*** *and* ***Multi expressions****.*

Example 5.8 (Graph Operation Expressions)**.** *The graph operation $(\mathtt{comp}^\mathcal{C} + \mathtt{comp}^\mathcal{C})/\varphi$ in Example 5.6 is represented by the Γ_{cat}-expression $\ulcorner[syn(\varphi)](\mathtt{comp}|\mathtt{comp})\urcorner$. By* ***Symbolic composition****, we obtain then the Γ_{cat}-expression $\ulcorner([syn(\varphi)](\mathtt{comp}|\mathtt{comp});syn(\varrho);\mathtt{comp})\urcorner$ representing the graph operation $((\mathtt{comp}^\mathcal{C} + \mathtt{comp}^\mathcal{C})/\varphi)\,;^\varrho \mathtt{comp}^\mathcal{C}$ in Example 5.7.*

The graph operation $(((\mathtt{comp}^{\mathcal{C}} + \pi^{\mathtt{I}}_{\mathtt{I}})/\psi)\,;^{\rho}\,\mathtt{comp}^{\mathcal{C}}$ *in Example 5.7 is represented by the* Γ_{cat}*-expression* $\ulcorner([syn(\psi)](\mathtt{comp}|(\mathtt{I},\mathtt{I}));syn(\rho);\mathtt{comp})\urcorner$ *while* $(((\pi^{\mathtt{I}}_{\mathtt{I}} + \mathtt{comp}^{\mathcal{C}})/\psi')\,;^{\rho'}\,\mathtt{comp}^{\mathcal{C}}$ *corresponds to the* Γ_{cat}*-expression* $\ulcorner([syn(\psi')]((\mathtt{I},\mathtt{I})|\mathtt{comp});syn(\rho');\mathtt{comp})\urcorner$*. Both* Γ_{cat}*-expressions share the same canonical arity span shown in Figure 13, and thus, we can express the associativity law (33) for the composition of morphisms in categories by an equation between* Γ_{cat}*-expressions:*

$$([syn(\psi)](\mathtt{comp}|(\mathtt{I},\mathtt{I}));syn(\rho);\mathtt{comp}) = ([syn(\psi')]((\mathtt{I},\mathtt{I})|\mathtt{comp});syn(\rho');\mathtt{comp}). \tag{34}$$

In (34), we do have a twofold-syntactic and semantic equality between graph operation expressions: equal arity and equal semantics. A future equational calculus for graph operation expressions should, however, also reflect arity renamings, as defined in Remark 4.1, and deal with semantic equality of graph operation expressions "up to arity renamings".

Generalizing the examples of the correspondence between graph operation expressions and graph operations in Example 5.8 and relying on the inductive definition of graph operation expressions in Definition 5.2, we now define "derived graph operations" as those graph operations that can be represented by graph operation expressions.

Definition 5.3 (Derived Graph Operations). *Let* $\Gamma = (OP, ar)$ *be a graph signature and* $\mathcal{G} = (\mathtt{G}, OP^{\mathcal{G}})$ *a* Γ*-algebra. For all* Γ*-expressions* $\ulcorner ge \urcorner \in GE(\Gamma)$ *with* $ar(ge) = \mathtt{I}_{ge} \xleftarrow{l_{ge}} \mathtt{B}_{ge} \xrightarrow{r_{ge}} \mathtt{O}_{ge}$ *,*
we can inductively define a map $ge^{\mathcal{G}} : \mathtt{G}^{\mathtt{I}_{ge}} \to \mathtt{G}^{\mathtt{O}_{ge}}$ *satisfying the commutativity condition in (16).* $ge^{\mathcal{G}}$ *is called the* derived graph operation *in* $\mathcal{G}$ represented by *ge.*

Projections: $(\mathtt{I},\mathtt{O})^{\mathcal{G}} := \pi^{\mathtt{I}}_{\mathtt{O}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$*, according to (17), for all* $\ulcorner(\mathtt{I},\mathtt{O})\urcorner \in GE(\Gamma)$.

Constants: $[\,]\mathtt{c}^{\mathcal{G}} := \pi^{\mathtt{I}}_{\mathtt{0}};\mathtt{c}^{\mathcal{G}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}_{\mathtt{c}}}$ *for all* $\ulcorner[\,]\mathtt{c}\urcorner \in GE(\Gamma)$.

$([\,]\mathtt{c}{\downarrow}\mathtt{O})^{\mathcal{G}} := \pi^{\mathtt{I}}_{\mathtt{0}};\mathtt{c}^{\mathcal{G}};\pi^{\mathtt{O}_{\mathtt{c}}}_{\mathtt{O}} : \mathtt{G}^{\mathtt{I}} \to \mathtt{G}^{\mathtt{O}}$ *for all* $\ulcorner([\,]\mathtt{c}{\downarrow}\mathtt{O})\urcorner \in GE(\Gamma)$.

Operations: $([syn(\varphi)]\mathtt{op})^{\mathcal{G}} := \mathtt{op}^{\mathcal{G}}/\varphi : \mathtt{G}^{\mathtt{I}'} \to \mathtt{G}^{\mathtt{O}'}$*, according to Section 5.3.2, for all* $\ulcorner[syn(\varphi)]\mathtt{op}\urcorner \in GE(\Gamma)$.

$([syn(\varphi)]\mathtt{op}{\downarrow}\mathtt{O})^{\mathcal{G}} := (\mathtt{op}^{\mathcal{G}}/\varphi);\pi^{\mathtt{O}'}_{\mathtt{O}} : \mathtt{G}^{\mathtt{I}'} \to \mathtt{G}^{\mathtt{O}}$ *for all* $\ulcorner([syn(\varphi)]\mathtt{op}{\downarrow}\mathtt{O})\urcorner \in GE(\Gamma)$.

Multi expressions: $(ge_1|\ldots|ge_k)^{\mathcal{G}} := (ge_1^{\mathcal{G}} + \ldots + ge_k^{\mathcal{G}})$*, according to Section 5.3.1, for all* $\ulcorner(ge_1|\ldots|ge_k)\urcorner \in GE(\Gamma)$.

$([syn(\varphi)](ge_1|\ldots|ge_k))^{\mathcal{G}} := (ge_1^{\mathcal{G}} + \ldots + ge_k^{\mathcal{G}})/\varphi$*, according to Section 5.3.1 and Section 5.3.2, for all* $\ulcorner[syn(\varphi)](ge_1|\ldots|ge_k)\urcorner \in GE(\Gamma)$.

Symbolic composition: $(ge_1;syn(\varrho);ge_2)^{\mathcal{G}} := ge_1^{\mathcal{G}};^{\varrho}ge_2^{\mathcal{G}} : \mathtt{G}^{\mathtt{I}_1} \to \mathtt{G}^{\mathtt{O}_2^{\varrho}}$*, according to Section 5.3.3, for all* $\ulcorner(ge_1;syn(\varrho);ge_2)\urcorner \in GE(\Gamma)$.

Hypothesis 5.3 (Substitutions Revisited). *In light of the discussion in Section 5.1.3, it may be possible to develop in the future a substitution calculus for graph operation expressions along the following lines: A* substitution *is given by a single- or multi-*Γ*-expression* $\ulcorner sub \urcorner$ *with arity*

$\mathtt{I}_1 \xleftarrow{l_1} \mathtt{B}_1 \xrightarrow{r_1} \mathtt{O}_1$ *. The substitution* $\ulcorner sub \urcorner$ *can be applied to a* Γ*-expression* $\ulcorner ge \urcorner$ *with arity*

$\mathtt{I}_2 \xleftarrow{l_2} \mathtt{B}_2 \xrightarrow{r_2} \mathtt{O}_2$ *if there is a graph isomorphism* $\varrho : \mathtt{I}_2 \to \mathtt{O}_1$. Substitution application *is then carried out in two steps: (1) We build the expression* $\ulcorner(sub;syn(\varrho);ge)\urcorner$ *through symbolic composition. If* $\ulcorner ge \urcorner$ *is not a basic* Γ*-expression, this expression will not be a* Γ*-expression in the sense of Definition 5.2! (2) We transform the expression* $\ulcorner(sub;syn(\varrho);ge)\urcorner$ *into an equivalent* Γ*-expression* $\ulcorner \widehat{ge} \urcorner$ *in the sense of Definition 5.2.*

To illustrate this idea, we outline for the graph signature Γ_{cat} *an example of an equivalence between a symbolic-composition expression* $\ulcorner(sub;syn(\varrho);ge)\urcorner$*, which is not a* Γ_{cat}*-expression in the sense of Definition 5.2, and a* Γ_{cat}*-expression* $\ulcorner \widehat{ge} \urcorner$*.*

To begin with a feasible visualization, we consider the Γ_{cat}*-expression* $\ulcorner[syn(\phi)](\mathtt{comp}|\mathtt{comp})\urcorner$ *where* ϕ *and the arity* $\mathtt{I}' \xhookleftarrow{l'} \mathtt{B}' \xhookrightarrow{r'} \mathtt{O}'$ *are described in Figure 14.*

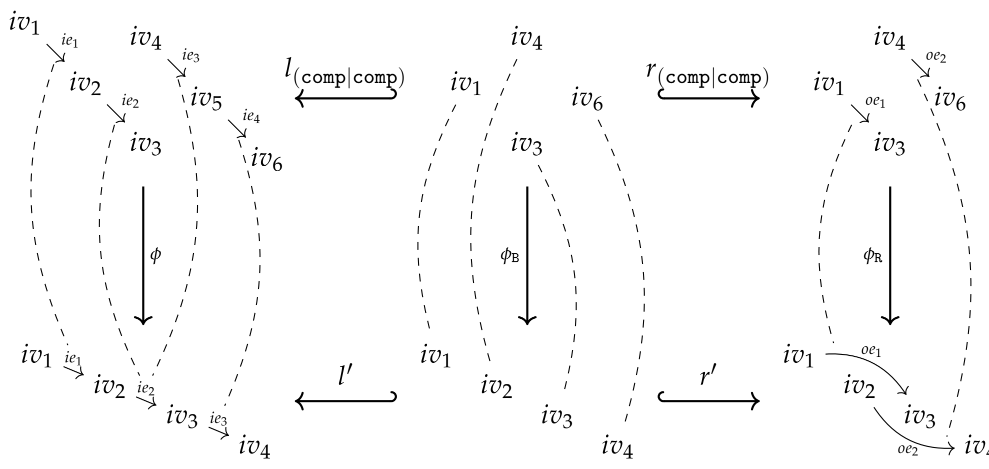

$l_{(\mathtt{comp}|\mathtt{comp})}$
$r_{(\mathtt{comp}|\mathtt{comp})}$
ϕ
$\phi_{\mathtt{B}}$
$\phi_{\mathtt{R}}$
l'
r'

Figure 14. Parallel application of two composition operations on overlapping pairs of arrows.

The sample substitution *is given by the extended* Γ_{cat}*-expression*

$$\ulcorner sub\urcorner := \ulcorner[syn(\bar{\phi})](\mathtt{comp}|\mathtt{comp}|(\mathtt{I},\mathtt{I})|(\mathtt{I},\mathtt{I}))\urcorner$$

with arity $ar(sub) = \mathtt{I}_{sub} \xhookleftarrow{l_{sub}} \mathtt{B}_{sub} \xhookrightarrow{r_{sub}} \mathtt{O}_{sub}$ *depicted in Figure 15.* $\mathtt{I}$ *is the canonical input arity graph* $iv_1 \xrightarrow{ie_1} iv_2$ *. The input arity of* $\ulcorner(\mathtt{comp}|\mathtt{comp}|(\mathtt{I},\mathtt{I})|(\mathtt{I},\mathtt{I}))\urcorner$ *extends the input arity of* $\ulcorner(\mathtt{comp}|\mathtt{comp})\urcorner$ *by the arrows* $iv_7 \xrightarrow{ie_5} iv_8 \qquad iv_9 \xrightarrow{ie_6} iv_{10}$ *while* $\bar{\phi}$ *extends* ϕ *by the assignments* $\{iv_7 \mapsto iv_1, ie_5 \mapsto ie_1, iv_8 \mapsto iv_2, iv_9 \mapsto iv_3, ie_6 \mapsto ie_3, iv_{10} \mapsto iv_4\}$*.*

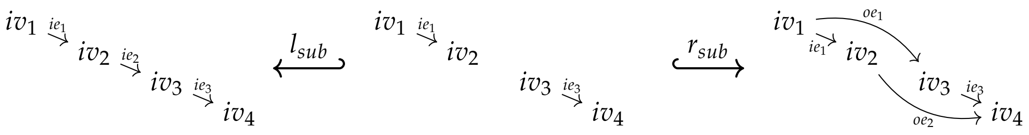


Figure 15. Arity of the Γ_{cat}-expression $\ulcorner sub\urcorner = \ulcorner[syn(\bar{\phi})](\mathtt{comp}|\mathtt{comp}|(\mathtt{I},\mathtt{I})|(\mathtt{I},\mathtt{I}))\urcorner$.

As a Γ_{cat}*-expression, we consider* $\ulcorner ge\urcorner := \ulcorner[syn(\varphi)](\mathtt{comp}|\mathtt{comp})\urcorner$ *with arity* $ar(ge) = \mathtt{I}_{ge} \xhookleftarrow{l_{ge}} \mathtt{B}_{ge} \xhookrightarrow{r_{ge}} \mathtt{O}_{ge}$ *shown in Figure 16.* φ *is given by the assignments* $\{ie_1 \mapsto ie_1, ie_2 \mapsto ie_2, ie_3 \mapsto ie_3, ie_4 \mapsto ie_4\}$*.*

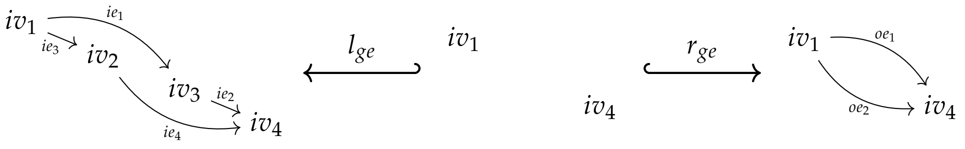


Figure 16. Arity of the Γ_{cat}-expression $\ulcorner ge\urcorner = \ulcorner[syn(\varphi)](\mathtt{comp}|\mathtt{comp})\urcorner$.

By means of the isomorphism $\varrho : \mathtt{I}_{ge} \to \mathtt{I}_{sub}$ *that is the identity on nodes, we can build the following symbolic-composition expression that is not a* Γ_{cat}*-expression*

$$\ulcorner(sub; syn(\varrho); ge)\urcorner = \ulcorner[syn(\bar{\phi})](\mathtt{comp}|\mathtt{comp}|(\mathtt{I},\mathtt{I})|(\mathtt{I},\mathtt{I})); syn(\varrho); [syn(\varphi)](\mathtt{comp}|\mathtt{comp})\urcorner$$

Using the two Γ_{cat}-expressions in (34), we can, however, construct a Γ_{cat}-expression $\ulcorner\widehat{ge}\urcorner$ given by

$$\ulcorner[syn(\theta)](([syn(\psi)](\texttt{comp}|(\texttt{I},\texttt{I}));syn(\rho);\texttt{comp})|([syn(\psi')]((\texttt{I},\texttt{I})|\texttt{comp});syn(\rho');\texttt{comp}))\urcorner$$

All four expressions share the same canonical input arity graph $\texttt{I}'$ in Figure 14, and θ is defined as the cotuple $\theta := [id_{\texttt{I}'}, id_{\texttt{I}'}] : \texttt{I}' + \texttt{I}' \to \texttt{I}'$.

$\ulcorner(sub;syn(\varrho);ge)\urcorner$ and $\ulcorner\widehat{ge}\urcorner$ do have the same arity and are equivalent, that is, for all Γ_{cat}-algebras $\mathcal{G} = (\texttt{G}, OP^{\mathcal{G}})$ we have $(sub;syn(\varrho);ge)^{\mathcal{G}} = \widehat{ge}^{\mathcal{G}}$.

Hypothesis 5.4 (Operations by Subgraphs)**.** *At the beginning of this section we discussed that any subgraph of a graph of graph terms defines a graph operation (see Lemma 5.1). A reasonable question is to what extent this method of defining graph operations can be simulated by means of "graph operation expressions" and "derived graph operations". We claim that it is possible to inductively prove a statement like the following:*

> *For any finite canonical input arity graph $\texttt{I}$ and any finite subgraph $\texttt{O} \sqsubseteq \texttt{T}_\Gamma(\texttt{I})$, there exist a graph operation expression $ge \in GE(\Gamma)$ with input arity $\texttt{I}$ and an output arity $\texttt{O}'$ isomorphic to $\texttt{O}$ such that for all Γ-algebras $\mathcal{G} = (\texttt{G}, OP^{\mathcal{G}})$, we have $ge^{\mathcal{G}} = \delta^{\texttt{I}}_{\texttt{O}}(\boldsymbol{b})$ (up to arity renaming) for the graph operation $\delta^{\texttt{I}}_{\texttt{O}} : \texttt{G}^{\texttt{I}} \to \texttt{G}^{\texttt{O}}$ defined in Lemma 5.1.*

There are two essential means that should allow us to prove such a statement:

1. *By means of the restriction expressions $\ulcorner([\,]\texttt{c}{\downarrow}\texttt{O})\urcorner$ and $\ulcorner([syn(\varphi)]\texttt{op}{\downarrow}\texttt{O})\urcorner$, we can simulate the "pseudo operation symbols" $\texttt{c}_{ov}$, $\texttt{c}_{oe}$, $\texttt{op}_{ov}$, $\texttt{op}_{oe}$ and their semantics by choosing $\texttt{O}$ to be the single output vertex ov or the single output edge oe (together with its source and target).*
2. *Once we have been able to represent all the single vertices and edges in a subgraph $\texttt{O} \sqsubseteq \texttt{T}_\Gamma(\texttt{I})$ by corresponding graph operation expressions, we can combine them into one graph operation expression using the trick used in Conjecture 5.3 to define the graph operation expression $\ulcorner\widehat{ge}\urcorner$. All the single output arities will be soldered together resulting in a graph isomorphic to $\texttt{O}$.*

6. Operations in Topoi

When we started to write the paper, we intended to round it up with a longer section to summarize the results and findings, and to lift them up to a more abstract level.

On the way, we discovered, however, too many new things and results, which we simply had to investigate and include in the paper, and thus it became a bit too long and overloaded. Therefore, we reserve a detailed categorical analysis of the results and findings of the paper and the development of a general theory of operations in topoi as a topic of future research. We already spent, however, some essential effort to revise traditional concepts and results, and to lift them up to a more categorical level; thus, we want to include, at least, some few remarks concerning this topic.

We are quite sure that all (!) the definitions, constructions, and results, including term algebras and the result that subalgebras are regular monic, can be generalized to presheaf topoi (C-sets) $[\mathsf{C} \to \mathsf{Set}]$ with C as a *simple category* in the sense of [22]. Simple categories are a special class of "finite categories with no cycles of non-identity morphisms" and play a role in the foundation of Homotopy Type Theory.

There is no problem defining signatures and algebras in arbitrary topoi, analogous to graph signatures and graph algebras. However, terms and term algebras will, in general, not be available. One of the main outcomes of the paper is that there is no need for term algebras to define derived operations. It is sufficient to define "operation expressions" and their semantics, and this can be performed in arbitrary topoi, as long as we drop the request that operation expressions should be represented "syntactically"!

We hope that even a prospective substitution calculus for graph operation expressions (along the ideas in Conjecture 5.3) can be generalized to operation expressions in arbitrary topoi.

7. Related Work

The work presented in this paper has its origins in the pioneering work on *generalized sketches* of a group around Zinovy Diskin in the 1990s [3–5]. The concept of *sketch operation* in [4] was the starting point for the joint paper [7] on graph operations and free graph algebras. As discussed in Section 4.2, it turned out, however, that the original definition of arities and graph operations in [7] was not appropriate to define *graph operation expressions* representing *derived graph operations* in analogy to the role of *terms* as representations of derived operations in traditional Universal Algebra.

Close to finalizing the paper and by a chain of accidents, we found out about a paper on graph operations (only available in French!) [6] from Albert Burroni, a former PhD student of Charles Ehresmann. The ambition of [6] is very much in accordance with the intentions behind our paper. Graph operations are also defined as maps from $\mathtt{G}^{\mathtt{I}}$ to $\mathtt{G}^{\mathtt{O}}$. There are, however, essential conceptual and technical differences between both papers:

- Input and output arity graphs are not described with concrete syntactic identifiers but are implicitly considered as being given by "equivalence up to arity renaming".
- There are no boundaries and thus no systematic treatment of "preservation conditions" as we formalized them by means of the commutativity condition in (16).
- In the examples, preservation conditions are expressed by means of ad hoc equations between vertices and/or edges, respectively.
- There is no explicit notion of "derived graph operation", and the issue of "graph operation expressions" is not addressed at all.
- Derived graph operations, in our sense, appear only implicitly when properties of graph operations are described by means of equations between vertices and/or edges.
- Apart from that, Ref. [6] demonstrates the usefulness and appropriateness of graph operations in category theory, and in particular, it seems to be worth investigating the relation between Chapter 2 and the first-order sketches introduced in [1] .

Thanks to the referees for drawing our attention to [23,24]. This is a work that should have been cited in [25] as an excellent underpinning of the claim that fibered structures provide an appropriate "technological space" where logical deduction can take place. We add here some comments on the relationship between the present paper and [23,24]:

- There are no graph operations but only plain algebraic operations in [23,24].
- However, the authors do not follow the tradition in Universal and Categorical Algebra of defining algebras as "indexed structures" [26] where sort symbols are interpreted by sets and operation symbols by maps from finite Cartesian products of sets into sets. Instead, they describe algebras as "fibred structures" utilizing so-called "m-graphs".
- In [23,24], the authors consider many-sorted algebraic signatures, while we restrict ourselves in the paper and in the following discussion to unsorted algebraic signatures Σ.
- m-graphs can encode the "graphs" of operations in traditional "indexed Σ-algebras". That is, each element of the graph $gr(\mathtt{op}^{\mathcal{A}}) := \{((a_1,\ldots,a_n),a) \mid \mathtt{op}^{\mathcal{A}}(a_1,\ldots,a_n) = a\}$ of an n-ary operation $\mathtt{op}^{\mathcal{A}} : A^n \to A$ in an "indexed Σ-algebra" $\mathcal{A}$ can be represented by a designated m-edge e with source $\mathsf{src}(e) = a_1 \ldots a_n$ and target $\mathsf{trg}(e) = a$.
- The discussion in Remark 4.4 concerning "graphs of graph operations" also applies to graphs of algebraic operations. There are no boundaries in the case of algebraic operations, and thus we have $\mathtt{R}_{\mathtt{op}} = \{i_1,\ldots,i_n,o\}$ for each n-ary operation symbol $\mathtt{op}$ in the signature Σ. Remark 4.4 proposes to represent the elements $((a_1,\ldots,a_n),a)$ in $gr(\mathtt{op}^{\mathcal{A}})$ by maps $\boldsymbol{a} : \{i_1,\ldots,i_n,o\} \to A$ with $\boldsymbol{a}(i_k) = a_k$ for all $1 \le k \le n$ and $\boldsymbol{a}(o) = a$. Due to the notational conventions in Section 2, we would, however, denote those mappings simply by $(n+1)$-tuples $(a_1,\ldots,a_n,a)$.
- The set of all pairs $(\mathtt{op},(a_1,\ldots,a_n,a))$ with $\mathtt{op}$ in Σ and $((a_1,\ldots,a_n),a) \in gr(\mathtt{op}^{\mathcal{A}})$ for an "indexed Σ-algebra" $\mathcal{A}$ constitutes the disjoint union of the graphs of all operations in $\mathcal{A}$ and can obviously serve as the set E of m-edges of an "m-graph encoding" of

the "indexed Σ-algebra" $\mathcal{A}$. The carrier set A of $\mathcal{A}$ is thereby the set of vertexes of the m-graph encoding of $\mathcal{A}$.

- For many-sorted signatures Σ we can construct m-graph encodings of "indexed Σ-algebras" that are fully analogous.
- It is obvious that the m-graph encodings of terminal "indexed Σ-algebras", where each sort in a many-sorted signature Σ is interpreted by a singleton, exactly correspond to the m-graphs serving as signatures in [23,24]. Thus, there is indeed a unique m-graph homomorphism from the m-graph encoding of any "indexed Σ-algebra" into the corresponding signature in the sense of [23,24].
- The "syntactic Lawvere theories" in our paper appear in [23,24] as finite product categories freely generated by m-graphs.

Since we use parallel and sequential composition of maps to define derived graph operations, there is of course a certain overlap with monoidal categories, term graphs [20] and string diagrams [21]. However, our approach to derived graph operations essentially deviates from traditional monoidal categories and string diagrams:

- Traditional monoidal categories and string diagrams only deal with single isolated items as the input and output of operations.
- This manifests itself in the absence of boundaries.
- The essential difference is that the presence of boundaries forces us to replace "internal copying of items in diagrams" with "soldering of input and output ports of diagrams", i.e., by constructing instances of diagrams as a whole.

It is an interesting and open question to what extent it might be useful and feasible to define categories with (equivalence classes of) graph operation expressions as morphisms. A more exotic question would be if those hypothetical categories can be characterized and axiomatized in analogy to the different kinds of monoidal categories and string diagrams.

8. Conclusions

One of the roles of *terms* in Universal Algebra is to represent *derived operations*, i.e., operations that can be built up from the basic operations in an algebra. Relying on a revised version of graph operations, we defined *graph operation expressions* as a counterpart to terms. We identified three basic mechanisms to construct new graph operations out of given ones: parallel composition, instantiation and sequential composition. These mechanisms allowed us to construct for all graph operation expressions a corresponding *derived graph operation* in any graph algebra.

In another direction, we made a first step toward "Universal Graph Algebra"; i.e., we generalized some basic model-theoretic concepts and results from algebras to graph algebras. In particular, we generalized the concept *generated subalgebra* and proved that all monomorphic homomorphisms between graph algebras are regular.

We made an overall and essential effort to present definitions, concepts, constructions, results, and proofs in a more categorical way such that it can be straightforwardly lifted to the level of topoi in the future.

Besides the missing proofs of the conjectures, there are many open ends and many things that could or should be carried out. In particular, an equational calculus for graph operations relying on the equality of graphs (and not of single vertices and/or edges) is very much demanded. Such a calculus is also expected to provide a basis to define an equivalence relation for graph operation expressions and, in turn, for a semantic-preserving rewriting of graph operation expressions.

Author Contributions: Writing-original draft, U.W. and T.T.T.; Writing—review and editing, U.W. and T.T.T.; Supervision, U.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We want to thank the guest editor of this special volume for encouraging us to write this paper and the anonymous referees for their helpful suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wolter, U. Logics of Statements in Context—Category Independent Basics. *Mathematics* **2022**, *10*, 1085.
2. Makkai, M. Generalized Sketches as a Framework for Completeness Theorems. *J. Pure Appl. Algebra* **1997**, *115*, 49–79, 179–212, 214–274. [CrossRef]
3. Cadish, B.; Diskin, Z. Heterogeneous View Integration via Sketches and Equations. In Proceedings of the ISMIS, Zakopane, Poland, 9–13 June 1996; Volume 1079, pp. 603–612.
4. Diskin, Z. *Databases as Diagram Algebras: Specifying Queries and Views Via the Graph-Based Logic of Sketches*; Technical Report 9602; Frame Inform Systems: Riga, Latvia, 1996.
5. Diskin, Z.; Kadish, B. A Graphical Yet Formalized Framework for Specifying View Systems. In Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, Nevsky Dialect, St. Petersburg, Russia, 2–5 September 1997; pp. 123–132.
6. Burroni, A. Algèbres graphiques (sur un concept de dimension dans les langages formels). *Cahiers de Topologie et Géométrie Différentielle Catégoriques* **1981**, *22*, 249 – 265.
7. Wolter, U.; Diskin, Z.; König, H. Graph Operations and Free Graph Algebras. In *Graph Transformation, Specifications, and Nets: In Memory of Hartmut Ehrig*; Heckel, R., Taentzer, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 313–331. [CrossRef]
8. Poigné, A. Algebra categorically. In *Category Theory and Computer Programming*; Pitt, D., Abramsky, S., Poigné, A., Rydeheard, D., Eds.; Springer: Berlin/Heidelberg, Germany, 1986; pp. 76–102. [CrossRef]
9. Ehrig, H.; Ehrig, K.; Prange, U.; Taentzer, G. *Fundamentals of Algebraic Graph Transformation*; Monographs in Theoretical Computer Science; An EATCS Series; Springer: Berlin/Heidelberg, Germany, 2006. [CrossRef]
10. Barr, M.; Wells, C. *Category Theory for Computing Science*; Series in Computer Science; Prentice Hall International: London, UK, 1990.
11. Pierce, B.C. *Basic Category Theory for Computer Scientists*; The MIT Press: Cambridge, MA, USA; London, UK, 1991.
12. Wolter, U. *Cogenerated Quotient Coalgebras*; Technical Report Report No 299; Department of Informatics, University of Bergen: Bergen, Norway, 2005.
13. Ehrig, H.; Mahr, B. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*; EATCS Monographs on Theoretical Computer Science 6; Springer: Berlin/Heidelberg, Germany, 1985.
14. Wechler, W. *Universal Algebra for Computer Scientists;* Monographs in Theoretical Computer Science; An EATCS Series; Springer: Berlin/Heidelberg, Germany, 1992; Volume 25, p. 339. [CrossRef]
15. MacLane, S. *Categories for the Working Mathematician*; Graduate Texts in Mathematics Volume 5; Springer: New York, NY, USA, 1978; pp. xii+318.
16. Goldblatt, R. *Topoi: The Categorial Analysis of Logic*; Dover Publications: New York, NY, USA, 1984.
17. Reichel, H. *Initial Computability, Algebraic Specifications, and Partial Algebras*; Oxford University Press: Oxford, UK, 1987.
18. Wolter, U. An Algebraic Approach to Deduction in Equational Partial Horn Theories. *J. Inf. Process. Cybern. EIK* **1990**, *27*, 85–128.
19. Claßen, I.; Große-Rhode, M.; Wolter, U. Categorical concepts for parameterized partial specifications. *Math. Struct. Comp. Sci.* **1995**, *5*, 153–188. [CrossRef]
20. Corradini, A.; Gadducci, F. An Algebraic Presentation of Term Graphs, via GS-Monoidal Categories. *Appl. Categ. Struct.* **1999**, *7*, 299–331.
21. Selinger, P. A Survey of Graphical Languages for Monoidal Categories. In *New Structures for Physics*; Coecke, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 289–355. [CrossRef]
22. Makkai, M. First Order Logic with Dependent Sorts, with Applications to Category Theory. Available online: https://www.math.mcgill.ca/makkai/folds/foldsinpdf/FOLDS.pdf (accessed on 17 September 2023).
23. Sernadas, A.; Sernadas, C.; Rasga, J.; Coniglio, M. A Graph-theoretic Account of Logics. *J. Log. Comput.* **2009**, *19*, 1281–1320.
24. Sernadas, A.; Sernadas, C.; Rasga, J.; Coniglio, M. On Graph-theoretic Fibring of Logics. *J. Log. Comput.* **2009**, *19*, 1321–1357. [CrossRef]
25. Wolter, U.; Martini, A.; Häusler, E. Indexed and fibered structures for partial and total correctness assertions. *Math. Struct. Comput. Sci.* **2022**, *32*, 1145–1175.
26. Wolter, U. Indexed vs. fibred structures—A field report. *Rom. J. Pure Appl. Math.* **2020**, *66*, 813–830.