Article

Logics of Statements in Context – First-Order Logic Files

Uwe Wolter [*]

Posted Date: 11 February 2025

doi: 10.20944/preprints202502.0785.v1

Keywords: first-order logic; open formula; logic of statements in context; abstract model theory; institution; institution comorphism; Grothendieck institution; substitution; sketch; elementary diagram

Article

Logics of Statements in Context—First-Order Logic Files

Uwe Wolter

Department of Informatics, University of Bergen, Norway; uwe.wolter@uib.no

Abstract: Logics of Statements in Context have been proposed as a general framework to describe and relate, in a uniform and unifying way, a broad spectrum of logics and specification formalisms which also comprise "open formulas". Especially, it has been shown that we can define arbitrary first-order "open formulas" in arbitrary categories. At present, there are two deficiencies. In the general case only signatures with predicate symbols are considered and institutions of statements in context are only defined for single signatures. In this paper we elaborate the special case of traditional many-sorted First-Order Logic. We show that any many-sorted first-order signature Σ with predicate and (!) operation symbols gives rise to an institution $\mathcal{FL}_\Sigma$ of Σ-statements in context, and that any signature morphism results in a comorphism between the corresponding institutions. We prove that we obtain a functor $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ from the category of signatures into the category of institutions and comorphisms. We construct a corresponding Grothendieck institution $\mathcal{FL}^\sharp$ and prove that $\mathcal{FL}^\sharp$ is indeed an extension of the traditional institution of First-Order Logic which only comprises "closed formulas". We also investigate substitutions in detail and discuss (elementary) diagrams in the sense of traditional First-Order Logic.

Keywords: first-order logic; open formula; logic of statements in context; abstract model theory; institution; institution comorphism; Grothendieck institution; substitution; sketch; elementary diagram

1. Introduction

Logics of Statements in Contexts have been introduced in [1] as a general framework enabling us to describe, analyze, relate and unify a variety of diverse concepts, results and constructions from a broad spectrum of logics and other formalisms including First-Order Logic [2–4], Universal Algebra [5], Algebraic Specifications [6,7], Ehresmann Sketches [8], Generalized Sketches [9–11], Description Logic [12] and Graph Transformations [13], for example.

From the more focused perspective of logic, Logics of Statements in Contexts propose a uniform way to define logics also comprising "open formulas". Especially, it has been shown that we can define arbitrary first-order "open formulas" in arbitrary categories.

At present, there are two main deficiencies. In the general case we do not consider operations since it is quite unclear for us how to generalize the traditional concept of operation for the category Set of sets and total maps to arbitrary categories. Some results have been achieved for graph operations, i.e., for operations in structures with directed multi graphs as carriers instead of sets [14,15]. We are convinced that these results can be extended to arbitrary categorical presheaves instead of the presheaf Graph. However, even in case of Graph we have not been able yet to come up with a reasonable substitution calculus. We adapted in [1] the concept of *institution* [16,17] as an adequate pattern to present the basic constituents of a logic. That we constructed in [1] only institutions of statements in context for single signatures, is the second main deficiency.

In this paper we turn back to the roots. We elaborate the paradigmatic special case of traditional many-sorted First-Order Logic where predicates and operations are combined. We show that any first-order signature Σ with predicate and (!) operation symbols gives rise to an institution $\mathcal{FL}_\Sigma$ of Σ-statements in context, and that any signature morphism transforms into a comorphism between the corresponding institutions. We prove that we obtain a functor $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ from the category of signatures into the category of institutions and comorphisms. We construct a corresponding Grothendieck institution $\mathcal{FL}^\sharp$ [17,18] and prove that $\mathcal{FL}^\sharp$ is indeed an extension of the traditional institution of first-order logic $\mathcal{FOL}$, as presented in [16,17] for example, which only comprises "closed formulas". In addition, it turns out that $\mathcal{FL}^\sharp$ is not only opfibred over Sig (since it is a Grothendieck institution) but even over $\mathcal{FOL}$ in a certain sense.

It is well-known that *empty sorts* potentially cause the problem that deduction rules for one-sorted first-order logic are, in general, not sound for many-sorted deduction [17,19]. We demonstrate that *contexts* are an appropriate tool to localize and handle this problem in the spirit of [19].

Another objective, leading us to the framework of Logics of Statements in Context, was the longing for a way of describing logics in a uniform abstract manner that is close to the way it is often done in traditional logic. We want to be allowed to restrict ourselves to finite signatures and to describe the language of a logic in an enumerable manner. We prefer a clear distinction between constant symbols and variables.

In addition, we intend to do (abstract) model theory even in case of finite signatures. From the perspective of model theory *contexts* serve as a bridge between syntax (constant symbols, variables) and semantics (elements of carrier sets), and can be utilized to define different kinds of *sketches* in analogy to the different kinds of diagrams in traditional First-Order Logic. In such a way, contexts also establish a "technological space" where deduction can take place. Deduction in Logics of Statements in Contexts is a big topic of its own and can not be covered in this paper.

To facilitate the reading and understanding of the paper, we recapitulate some methodological principles which guided the design of Logics of Statements in Context:

- "Open formulas" should be treated as first-class citizens.
- All syntactic entities and not only the basic ones, like predicate and operation symbols, should have a semantics in a given structure.
- The semantics of a term in a given structure is a "derived operation" assembled from the "basic operations", i.e., the semantics of operation symbols in the given structure. So, terms give us a syntactic representation of "derived operations" at hand.
- To obtain, in the many-sorted case, a sound and complete deduction calculus it is not enough to take into account only the set of all free variables syntactically appearing in a formula. The set of free variables needs to be bound, in principle, to a bigger context.
- "Open formulas" without such a binding are called *expressions*. We do not consider expressions as formulas but, in analogy to terms, as syntactic representations of "derived predicates" assembled from the "basic predicates", i.e., the semantics of predicate symbols in the given structure.
- Expressions bound to a context are called *statements in context*.
- In such a way, we can also encapsulate the relatively intricate construction of first-order expressions, in the sense, that the translation of statements along context morphisms can be simply defined by composing bindings with context morphisms. We adapted this "encapsulation trick" from [16] where it is used for initial and free constraints.

The paper is organized as follows. Section 2 recapitulates some basic concepts and corresponding notational conventions. Section 3 gives a rigorous and novel description of basic concepts in many-sorted first-order logic and their semantics – signatures, terms and expressions.

Section 4 is devoted to signature morphisms. We start with a detailed analysis of the functors induced by maps between finite sets of sort symbols and elucidate, especially, the special role of disjoint sorted sets. This enables us to describe the translation of terms and expressions along signature morphisms by means of natural transformations. On the semantic side, we define reducts of structures, homomorphisms, derived operations and derived predicates in the opposite direction. This gives us, finally, a *satisfaction condition for expressions and assignments* at hand. The traditional institution of first-order logic $\mathcal{FOL}$ is reconstructed by restraining the natural transformations and the satisfaction condition to closed expressions.

In Section 5 we finally present the institutions $\mathcal{FL}_\Sigma$ of Σ-statements in context, the functor $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ and the Grothendieck institution $\mathcal{FL}^\sharp$. In Section 6 we investigate substitutions in detail and develop corresponding extensions of constructions and results in Section 3 and 4. Section 7 recalls the concepts sketch and sketch implication. We adapt the proposal in [1] to formalize diagrams, in the sense of traditional First-Order Logic [2,3], as sketches. We conclude the paper with Section 8.

2. Notations and Preliminaries

Categories and Functors $|\mathsf{C}|$ denotes the collection of objects of a category C and C_{Mor} 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.

An object $\mathbf{0}$ in a category C is called *initial* if there exist for any object A in C a unique morphism from $\mathbf{0}$ into A often denoted by $!_A : \mathbf{0} \to A$ and called *initial morphism* for A.

$\mathsf{C} \sqsubseteq \mathsf{D}$ states that category C is a subcategory of category D. The corresponding *inclusion functor* is denoted by $\mathbf{In}_{\mathsf{C},\mathsf{D}} : \mathsf{C} \hookrightarrow \mathsf{D}$.

A category C is *small* if the collection C_{Mor}, and thus also the collection $|\mathsf{C}|$, is a set. Cat is the category of all small categories. Set denotes the category of all sets and all (total) maps. Cat and Set are not small! A category C is *locally small* if $\mathsf{C}(A,B)$ is a set for all objects A and B in C. Cat and Set are locally small! CAT is the category with all small categories and categories like Cat and Set as objects.

To lighten notation, we often use, in case $\mathbf{F}(\mathsf{C}') \subseteq \mathsf{D}'$, the same identifier $\mathbf{F} : \mathsf{C}' \to \mathsf{D}'$ for the restriction of a functor $\mathbf{F} : \mathsf{C} \to \mathsf{D}$ to subcatgories $\mathsf{C}' \sqsubseteq \mathsf{C}$, $\mathsf{D}' \sqsubseteq \mathsf{D}$.

For a category C we denote by $gr(\mathsf{C})$ the underlying *reflexive graph*, i.e., in $gr(\mathsf{C})$ we simply forget that there is a composition operation in C. Obviously, each functor $\mathbf{F} : \mathsf{C} \to \mathsf{D}$ comprises a corresponding *reflexive graph homomorphism* $\mathbf{F} : gr(\mathsf{C}) \to gr(\mathsf{D})$ between the underlying reflexive graphs (compare the concept of a "model of a graph in a category" in [8]). Be aware, however, that not every reflexive graph homomorphism $\mathbf{G} : gr(\mathsf{C}) \to gr(\mathsf{D})$ establishes also a functor $\mathbf{G} : \mathsf{C} \to \mathsf{D}$ because a reflexive graph homomorphism $\mathbf{G} : gr(\mathsf{C}) \to gr(\mathsf{D})$ is not required to be compatible with composition.

Natural Transformations and Functor Categories: If $\alpha : \mathbf{F} \Rightarrow \mathbf{G} : \mathsf{A} \to \mathsf{B}$ and $\beta : \mathbf{G} \Rightarrow \mathbf{H} : \mathsf{A} \to \mathsf{B}$ are natural transformations, the vertical composition of α and β is denoted $\alpha;\beta : \mathbf{F} \Rightarrow \mathbf{H} : \mathsf{A} \to \mathsf{B}$ such that for each $A \in |\mathsf{A}|$, $(\alpha;\beta)_A := \alpha_A;\beta_A$.

Also, if $\mathbf{F} : \mathsf{A} \to \mathsf{B}$, $\mathbf{G}, \mathbf{G}' : \mathsf{B} \to \mathsf{C}$, $\mathbf{H} : \mathsf{C} \to \mathsf{D}$ are functors and $\alpha : \mathbf{G} \Rightarrow \mathbf{G}' : \mathsf{B} \to \mathsf{C}$ is a natural transfomation, the horizontal compositions of $\mathbf{F}$ with α, and α with $\mathbf{H}$ are represented as $\mathbf{F}.\alpha : \mathbf{F};\mathbf{G} \Rightarrow \mathbf{F};\mathbf{G}' : \mathsf{A} \to \mathsf{C}$ and $\alpha.\mathbf{H} : \mathbf{G};\mathbf{H} \Rightarrow \mathbf{G}';\mathbf{H} : \mathsf{B} \to \mathsf{D}$ such that for each $C \in |\mathsf{C}|$, $A \in |\mathsf{A}|$, $(\mathbf{F}.\alpha)_C := \alpha_{\mathbf{F}(C)}$ whereas $(\alpha.\mathbf{H})_A := \mathbf{H}(\alpha_A)$.

$$\begin{tikzcd}
\mathsf{A} \arrow[r, "\mathbf{F}"] & \mathsf{B} \arrow[r, bend left, "\mathbf{G}"] \arrow[r, bend right, "\mathbf{G}'"'] \arrow[r, phantom, "\Downarrow\alpha"] & \mathsf{C} \arrow[r, "\mathbf{H}"] & \mathsf{D}
\end{tikzcd}$$

Moreover, we have for functors $\mathbf{F} : \mathsf{A} \to \mathsf{B}$, $\mathbf{G} : \mathsf{B} \to \mathsf{C}$, $\mathbf{H}, \mathbf{H}', \mathbf{H}'' : \mathsf{C} \to \mathsf{D}$, $\mathbf{I} : \mathsf{D} \to \mathsf{E}$ and natural transformations $\alpha : \mathbf{H} \Rightarrow \mathbf{H}'$, $\beta : \mathbf{H}' \Rightarrow \mathbf{H}''$ the following laws

$$\begin{tikzcd}
\mathsf{A} \arrow[r, "\mathbf{F}"] & \mathsf{B} \arrow[r, "\mathbf{G}"] & \mathsf{C} \arrow[r, bend left=50, "\mathbf{H}"] \arrow[r] \arrow[r, bend right=50, "\mathbf{H}''"'] \arrow[r, phantom, shift left=3ex, "\Downarrow\alpha"] \arrow[r, phantom, shift right=3ex, "\Downarrow\beta"] & \mathsf{D} \arrow[r, "\mathbf{I}"] & \mathsf{E}
\end{tikzcd}$$

$$(\mathbf{G}.\alpha).\mathbf{I} = \mathbf{G}.(\alpha.\mathbf{I}) \quad \text{and} \quad (\alpha;\beta).\mathbf{I} = (\alpha.\mathbf{I});(\beta.\mathbf{I}) \tag{1}$$

$$(\mathbf{F};\mathbf{G}).\alpha = \mathbf{F}.(\mathbf{G}.\alpha) \quad \text{and} \quad \mathbf{G}.(\alpha;\beta) = (\mathbf{G}.\alpha);(\mathbf{G}.\beta) \tag{2}$$

For categories C and D we denote by $[\mathsf{C} \to \mathsf{D}]$ or D^{C} the corresponding *functor category* with objects all functors from C into D, morphisms all natural transformations between those functors and vertical composition of natural transformations.

The traditional definition of natural transformations $\alpha : \mathbf{F} \Rightarrow \mathbf{G} : \mathsf{A} \to \mathsf{B}$ between functors can be reused to define natural transformations $\alpha : \mathbf{F} \Rightarrow \mathbf{G} : gr(\mathsf{A}) \to gr(\mathsf{B})$ for all (!) reflexive graph

doi:10.20944/preprints202502.0785.v1

homomorphisms between the underlying reflexive graphs of categories. The equations in (1) and (2) remain valid for those reflexive graph homomorphisms and natural transformations between them.

***S*-sorted Sets:** We can consider any set S as a discrete category. The corresponding functor category Set^S has S-indexed families $A = (A_s \mid s \in S)$ of sets as objects, called *S-sorted sets* or *S-sets* for short, and as morphisms $f : A \to B$ S-indexed families $(f_s : A_s \to B_s \mid s \in S)$ of maps, called *S-maps*. The *empty S-set* $\mathbf{0}_S := (\emptyset \mid s \in S)$ is the initial object in Set^S. $A = (A_s \mid s \in S)$ is called a *singleton S-set* if there is only one $s \in S$ with $A_s \neq \emptyset$ and if A_s is a singleton for this $s \in S$. Set^S_d denotes the full subcategory of Set^S given by all disjoint S-sets $A = (A_s \mid s \in S)$, i.e., we have $A_{s_1} \cap A_{s_2} = \emptyset$ for all $s_1, s_2 \in S$ with $s_1 \neq s_2$.

Inclusions: For any inclusion $A \subseteq B$ of sets there is a corresponding *inclusion map* $in_{A,B} : A \hookrightarrow B$ with $in_{A,B}(a) = a$ for all $a \in A$. There is an S-inclusion $A \subseteq B$ between S-sets $A = (A_s \mid s \in S)$ and $B = (B_s \mid s \in S)$ if $A_s \subseteq B_s$ for all $s \in S$. The corresponding inclusion S-map is $in_{A,B} := (in_{A_s,B_s} \mid s \in S) : A \hookrightarrow B$. Intersections, unions and complements of S-sets are also defined component wise.

***S*-typed Sets:** For any set S the slice category Set/S has as objects *S-typed sets*, i.e., pairs (C, t) of a set C and a *typing map* $t : C \to S$. A morphism $f : (C, t) \to (D, r)$ is given by a map $f : C \to D$ such that $f; r = t$.

The assignments $(C, t) \mapsto (t^{-1}(s) \mid s \in S)$ extend to a functor $\mathbf{I}_S : \mathsf{Set}/S \to \mathsf{Set}^S$ which restricts to an isomorphism $\mathbf{I}_S : \mathsf{Set}/S \to \mathsf{Set}^S_d$ where Set^S_d is the full subcategory of Set^S given by all disjoint S-sets. The inverse functor $\mathbf{S}_S : \mathsf{Set}^S_d \to \mathsf{Set}/S$ is given by union of component sets and assigns to each S-set $A = (A_s \mid s \in S)$ the S-typed set $(\bigcup A, t_A)$ with $t_A(a) = s$ if $a \in A_s$. $\mathbf{S}_S : \mathsf{Set}^S_d \to \mathsf{Set}/S$ can be extended to a functor $\mathbf{S}_S : \mathsf{Set}^S \to \mathsf{Set}/S$ by utilizing disjoint union (sum) instead of union for all non-disjoint S-sets. This establishes an equivalence $\mathbf{S}_S \dashv \mathbf{I}_S : \mathsf{Set}^S \to \mathsf{Set}/S$ of categories with $\mathbf{I}_S; \mathbf{S}_S = id_{\mathsf{Set}/S}$.

3. Basic Concepts and Constructions in First-Order Logic

Name spaces: It is under-communicated that the definition a morphism between concrete institutions often relies on the implicit assumption that the syntactic entities of both institutions are build upon the same stocks of symbol identifiers. We restrict ourselves in this paper to finitary syntactic entities. Therefore, we assume that all considered first-order institutions are build upon a certain choice of four enumerable and disjoint *name spaces*:

1. A set N_{fun} of names for operation symbols.
2. A set N_{pred} of names for predicate symbols.
3. A set N_{sort} of names for sort symbols which is equipped with a fixed total order.
4. A set N_{var} of names for variables which is equipped with a fixed total order.

In this paper we choose N_{sort} to be a subset of words with at least two letters over the Latin alphabet with lexicographic order, and for N_{var} we choose the totally ordered set $\{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots\}$.

3.1. Signatures and Structures

In this paper about first-order logic we use the traditional term "signature" instead of the neologism "footprint" coined for the general setting in [1]. Since we will neither misuse constant symbols to encode variables nor to encode elements of carriers of first-order structures, we can restrict ourselves to finite signatures only.

To define a many-sorted signature Σ, we first have to choose a finite set $S \subset N_{sort}$ of sort symbols. In [1] we require that the arities of predicate and operation symbols are defined by means of "variable declarations". For a chosen set S the corresponding "variable declarations", in the sense of [1], are given by the set Var_S of all finite and disjoint (!) S-sets $X = (X_s \mid s \in S)$ with $X_s \subset N_{var}$ for all $s \in S$. We will simply call them "S-sets of variables" instead of "variable declarations".

Definition 1 (Signature). *A **(many-sorted) signature** $\Sigma = (S, P, F, ar, in, out)$ is given by*

- *a finite set $S \subset \mathrm{N}_{sort}$ of sort symbols,*
- *a finite set $P \subset \mathrm{N}_{pred}$ of predicate (relation) symbols,*
- *a finite set $F \subset \mathrm{N}_{fun}$ of operation (function) symbols, and*
- *arity functions $ar : P \to Var_S$, $in : F \to Var_S$, $out : F \to Var_S$ where $out(\mathtt{op})$ is a singleton S-set for all $\mathtt{op} \in F$. By $srt(\mathtt{op})$ we denote the only $s \in S$ with $out(\mathtt{op})_s \neq \emptyset$. We also assume that $in(\mathtt{op})$ and $out(\mathtt{op})$ are disjoint for all $\mathtt{op} \in F$ (compare [15]).*

Remark 1 (Representation and Syntactification of S-sets of Variables). *Since N_{sort} is by assumption equipped with a total order, any finite set $S \subset \mathrm{N}_{sort}$ inherits from N_{sort} a total order thus we can represent, especially in examples, any S-set as an m-tuple of sets where $m = |S|$ is the* cardinality *of S.*

Moreover, for any S-set $X = (X_s \mid s \in S)$ of variables the union $\bigcup X$ of its components sets inherits a total order from N_{var}. This allows us to represent X by encoding the equivalent S-typed set $\mathbf{S}_S(X) = (\bigcup X, t_X)$ as an n-tuple $(v_1^X\!: s_1, \ldots, v_n^X\!: s_n)$ of variable declarations *with $n = |\bigcup X|$, v_i^X denoting the i-th variable in the totally ordered set $\bigcup X$ and $s_i := t_X(v_i^X)$ for all $1 \leq i \leq n$.*

In such a way, we can also syntactically encode the S-set X of variables by the string of symbols $\mathbf{syn}(X) := [v_1^X\!: s_1, \ldots, v_n^X\!: s_n]$.

Example 1 (Signature). *We chose $S = \{date < nat < prs\}$ with sort symbols "prs" for person and "nat" for natural number. The sample signature $\Sigma = (S, P, F, ar, in, out)$ is then defined by*

- *$P = \{\mathtt{fem}, \mathtt{par}, \mathtt{bef}\}$ with predicate symbols $\mathtt{par}$ for "parent", $\mathtt{fem}$ for "female" and $\mathtt{bef}$ for "before", and the following arities: $ar(\mathtt{fem}) = (\emptyset, \emptyset, \{x_1\})$ represented by $(x_1\!: prs)$, $ar(\mathtt{par}) = (\emptyset, \emptyset, \{x_1, x_2, x_3\})$ represented by $(x_1\!: prs, x_2\!: prs, x_3\!: prs)$ and $ar(\mathtt{bef}) = (\{x_1, x_2\}, \emptyset, \emptyset)$ represented by $(x_1\!: date, x_2\!: date)$; and*
- *$F = \{\mathtt{sp}, \mathtt{one}, \mathtt{born}, \mathtt{pl}\}$ with operation symbols $\mathtt{sp}$ for "sp(ecial day)", $\mathtt{pl}$ for "pl(us that many days)" and the following arities: $in(\mathtt{sp}) = in(\mathtt{one}) = \mathbf{0}_S = (\emptyset, \emptyset, \emptyset)$ represented by $(\,)$, $in(\mathtt{born}) = (\emptyset, \emptyset, \{x_1\})$ represented by $(x_1\!: prs)$, $in(\mathtt{pl}) = (\{x_1\}, \{x_2\}, \emptyset)$ represented by $(x_1\!: date, x_2\!: nat)$ and $out(\mathtt{sp}) = (\{y_1\}, \emptyset, \emptyset)$ represented by $(y_1\!: date)$, $out(\mathtt{one}) = (\emptyset, \{y_1\}, \emptyset)$ represented by $(y_1\!: nat)$, $out(\mathtt{born}) = out(\mathtt{pl}) = (\{y_1\}, \emptyset, \emptyset)$ represented by $(y_1\!: date)$.*

Now, we are going to define the category of structures for a given signature Σ.

Definition 2 (Structure). *For a signature $\Sigma = (S, P, F, ar, in, out)$ a **Σ-Structure** $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ is given by*

- *an S-set A, called the* carrier *of $\mathcal{A}$,*
- *a family $P^{\mathcal{A}} = \{\mathtt{p}^{\mathcal{A}} \mid \mathtt{p} \in P\}$ of sets $\mathtt{p}^{\mathcal{A}} \subseteq A^{ar(\mathtt{p})}$ of* valid assignments *for the predicate symbol $\mathtt{p}$ in $\mathcal{A}$, and*
- *a family $F^{\mathcal{A}} = \{\mathtt{op}^{\mathcal{A}} \mid \mathtt{op} \in F\}$ of operations (functions) $\mathtt{op}^{\mathcal{A}} : A^{in(\mathtt{op})} \to A^{out(\mathtt{op})}$ in $\mathcal{A}$.*

Remark 2 (Representation of Variable Assignments). *We consider an S-set $X = (X_s \mid s \in S)$ in Var_S with $n = |\bigcup X|$. For a given S-set A, S-maps $\boldsymbol{a} : X \to A$ will be also called* (variable) assignments of X in A.

Relying on Remark 1, we can establish a bijection between the set A^X of all assignments of X in A and the Cartesian product $A_{s_1} \times \ldots \times A_{s_n}$ with $s_i = t_X(v_i^X)$ for all $1 \leq i \leq n$ where v_i^X denotes the i-th variable in the totally ordered set $\bigcup X$. That is, any assignment *$\boldsymbol{a} : X \to A$ can equivalently be represented as an n-tuple $(a_1, \ldots, a_n) \in A_{s_1} \times \ldots \times A_{s_n}$ with $a_i = \boldsymbol{a}_{s_i}(v_i^X)$ for all $1 \leq i \leq n$. We will utilize both notations $\boldsymbol{a} : X \to A$ and $\boldsymbol{a} = (a_1, \ldots, a_n)$ synonymously.*

Note, that in case of the empty S-set $X = \mathbf{0}_S$ the only element $!_A : \mathbf{0}_S \to A$ in $A^{\mathbf{0}_S}$ is represented by the empty tuple $()$.

Example 2 (Structure). *For the sample signature Σ in Example 1 we describe a sample Σ-structure $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ thereby relying on the conventions in Remark 1 and 2:*

- *For the carrier $A = (A_{date}, A_{nat}, A_{prs})$ we choose A_{date} to be the set of all dates from 01.01.1700 on. A_{nat} is the set $\mathbb{N}^+$ of all positive natural numbers and A_{prs} is the set of all persons born in Europe from 01.01.1900 on.*
- *For the family $P^{\mathcal{A}} = \{\mathtt{par}^{\mathcal{A}}, \mathtt{fem}^{\mathcal{A}}, \mathtt{bef}^{\mathcal{A}}\}$ the predicate $\mathtt{par}^{\mathcal{A}} \subseteq A^{ar(\mathtt{par})}$ is given by all triples (p_1, p_2, p_3) of persons in A_{prs} such that p_2 is the father and p_3 is the mother of p_1. Note, that there is no information in $\mathtt{par}^{\mathcal{A}}$ about the parents of a person, if one of the parents is born before 01.01.1900 or outside Europe. $\mathtt{fem}^{\mathcal{A}}$ is the subset of all female persons in A_{prs} and $\mathtt{bef}^{\mathcal{A}}$ is given by the usual irreflexive before-relation between dates.*
- *For the family $F^{\mathcal{A}} = \{\mathtt{sp}^{\mathcal{A}}, \mathtt{one}^{\mathcal{A}}, \mathtt{born}^{\mathcal{A}}, \mathtt{pl}^{\mathcal{A}}\}$ the operation $\mathtt{born}^{\mathcal{A}} : A^{(x_1:prs)} \to A_{date}$ provides for each person p in A_{prs} the date of its day of birth. Note, that we have chosen A_{prs} and A_{date} in such way that $\mathtt{born}^{\mathcal{A}}$ is ensured to be a total operation! For any date $d \in A_{date}$ and any natural number $n \in A_{nat}$ $\mathtt{pl}^{\mathcal{A}}(d, n)$ is the date of the n-th day after d. $\mathtt{one}^{\mathcal{A}}()$ is the natural number 1 and $\mathtt{sp}^{\mathcal{A}}()$ the special date 19.10.1958.*

Homomorphisms preserve as well "truth" as "operation behavior".

Definition 3 (Homomorphism)**.** *A Σ-**homomorphism** $\varsigma : \mathcal{A} \to \mathcal{B}$ between two Σ-structures $\mathcal{A}$ and $\mathcal{B}$ is given by an S-map $\varsigma \colon A \to B$ such that*

1. *$\boldsymbol{a} \in \mathtt{p}^{\mathcal{A}}$ implies $\boldsymbol{a};\varsigma \in \mathtt{p}^{\mathcal{B}}$ for all $\mathtt{p} \in P$ and all assignments $\boldsymbol{a} : ar(\mathtt{p}) \to A$, i.e., the map $(\_;\varsigma) : A^{ar(\mathtt{p})} \to B^{ar(\mathtt{p})}$ restricts to a map $(\_;\varsigma) : \mathtt{p}^{\mathcal{A}} \to \mathtt{p}^{\mathcal{B}}$.*

$$\begin{tikzcd}
 & ar(\mathtt{p}) \arrow[dl, "\boldsymbol{a}\in\mathtt{p}^{\mathcal{A}}"'] \arrow[dr, "\boldsymbol{a};\varsigma\in\mathtt{p}^{\mathcal{B}}"] \arrow[d, phantom, "implies"] & \\
A \arrow[rr, "\varsigma"] & & B
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathtt{p}^{\mathcal{A}} \arrow[d, "(\_;\varsigma)"'] \arrow[r, phantom, "\subseteq"] & A^{ar(\mathtt{p})} \arrow[d, "(\_;\varsigma)"] \\
\mathtt{p}^{\mathcal{B}} \arrow[r, phantom, "\subseteq"] & B^{ar(\mathtt{p})}
\end{tikzcd}$$

2. *$\mathtt{op}^{\mathcal{A}}(\boldsymbol{a});\varsigma = \mathtt{op}^{\mathcal{B}}(\boldsymbol{a};\varsigma)$ for all $\mathtt{op} \in F$ and all input assignments $\boldsymbol{a} : in(\mathtt{op}) \to A$, i.e., $\mathtt{op}^{\mathcal{A}};(\_;\varsigma) = (\_;\varsigma);\mathtt{op}^{\mathcal{B}}$.*

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

$\mathsf{Str}(\Sigma)$ denotes the category of all Σ-structures and all Σ-homomorphisms.

3.2. Terms and First-Order Expressions: Syntax

We consider terms and first-order expressions as syntactic entities and define them as finite strings of symbols. To distinguish terms from meta-level expressions, such as $\mathtt{op}^{\mathcal{A}}(a_1, \ldots, a_n)$, we will use angle bracket symbols "$\langle$","$\rangle$", instead of parenthesis "(",")", 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.

To be prepared for later and future developments we introduce terms not only for S-sets of variables but for arbitrary disjoint (!) S-sets. The following is a traditional inductive definition of terms similar to [5,6] and relies on the conventions in Remark 2:

Definition 4 (Terms: Syntax)**.** *The S-set $T_\Sigma(A)$ of all Σ-**terms** over a disjoint S-set A is the smallest S-set of strings of symbols such that*

Variables: *$\ulcorner a \urcorner \in T_\Sigma(A)_s$ for all $a \in A_s$, $s \in S$;*

Constants: $\ulcorner\mathtt{c}\langle\rangle\urcorner \in T_\Sigma(A)_{srt(\mathtt{c})}$ *for all* $\mathtt{c} \in F$ *with* $in(\mathtt{c}) = \mathbf{0}_S$*, i.e.,* $|\bigcup in(\mathtt{op})| = \emptyset$*;*
Operations: $\ulcorner\mathtt{op}\langle t_1, \ldots, t_n\rangle\urcorner \in T_\Sigma(A)_{srt(\mathtt{op})}$ *for all* $\mathtt{op} \in F$ *with* $n = |\bigcup in(\mathtt{op})| \geq 1$ *and all assignments* $\boldsymbol{t} = (\ulcorner t_1\urcorner, \ldots, \ulcorner t_n\urcorner)$ *in* $T_\Sigma(A)^{in(\mathtt{op})}$*.*

Note that the assignments $a \mapsto \ulcorner a\urcorner$, assigning to each element in A the string constituted exactly by this single symbol, define an injective S-map $\eta_A\colon A \to T_\Sigma(A)$.

Note further, that each operation symbol $\mathtt{op} \in F$ with $n = |\bigcup in(\mathtt{op})| \geq 1$ is reborn as the Σ-term $\ulcorner\mathtt{op}\langle v_1^{in(\mathtt{op})}, \ldots, v_n^{in(\mathtt{op})}\rangle\urcorner \in T_\Sigma(in(\mathtt{op}))_{srt(\mathtt{op})}$ with the $v_i^{in(\mathtt{op})}$ as in Remark 2.

Remark 3 (Terms: Disjointness and Inclusions)**.** *The S-set $T_\Sigma(A)$ is disjoint since A is disjoint and since our concept of signature avoids the so-called "overloading of operation symbols"!*

In such a way, the inductive definition of $T_\Sigma(A)$ can be equivalently seen as the inductive definition of a set $\bigcup T_\Sigma(A)$ of strings together with a typing map $t_{T_\Sigma(A)} : \bigcup T_\Sigma(A) \to S$!

We do have $T_\Sigma(A) \subseteq T_\Sigma(B)$ whenever $A \subseteq B$. For a disjoint S-set B not every Σ-term in $T_\Sigma(B)$ "syntactically contains" all the elements in B. The S-set of all the elements in B, "syntactically appearing" in a Σ-term $t \in T_\Sigma(B)_s$, is the smallest S-subset $A \subseteq B$ such that $t \in T_\Sigma(A)_s$.

Example 3 (Terms: Syntax)**.** *There is no recursion in the sample signature Σ in Example 1 thus we have "up to renaming of variables" only five Σ-terms, all of sort date, in addition to the four "reborn operation symbols" $\ulcorner\mathtt{sp}\langle\,\rangle\urcorner$, $\ulcorner\mathtt{one}\langle\,\rangle\urcorner$, $\ulcorner\mathtt{born}\langle x_1\rangle\urcorner$ and $\ulcorner\mathtt{pl}\langle x_1, x_2\rangle\urcorner$. The five Σ-terms are $\ulcorner\mathtt{pl}\langle\mathtt{sp}\langle\,\rangle, \mathtt{one}\langle\,\rangle\rangle\urcorner$ over $\mathbf{0}_S$, $\ulcorner\mathtt{pl}\langle x_1, \mathtt{one}\langle\,\rangle\rangle\urcorner$ over $(x_1\colon date)$, $\ulcorner\mathtt{pl}\langle\mathtt{sp}\langle\,\rangle, x_1\rangle\urcorner$ over $(x_1\colon nat)$, $\ulcorner\mathtt{pl}\langle\mathtt{born}\langle x_1\rangle, \mathtt{one}\langle\,\rangle\rangle\urcorner$ over $(x_1\colon prs)$ and $\ulcorner\mathtt{pl}\langle\mathtt{born}\langle x_1\rangle, x_2\rangle\urcorner$ over $(x_1\colon prs, x_2\colon nat)$.*

Remark 4 (Syntactification of Substitutions)**.** *Variable assignments of the form $\boldsymbol{t} : X \to T_\Sigma(Y)$ will be also called **Σ-substitutions**. Due to Remark 2, we can uniquely represent any Σ-substitution $\boldsymbol{t} : X \to T_\Sigma(Y)$ by an n-tuple $(t_1, \ldots, t_n)$ of terms where $n = |\bigcup X|$.*

What we implicitly did in Definition 4, was to introduce the corresponding syntactic encoding $\mathbf{syn}(\boldsymbol{t}) := \ulcorner\langle t_1, \ldots, t_n\rangle\urcorner$ of the substitution $\boldsymbol{t} : X \to T_\Sigma(Y)$ as a string of symbols. For the injective S-map $\eta_X\colon X \to T_\Sigma(X)$ we get $\mathbf{syn}(\eta_X) := \ulcorner\langle v_1^X, \ldots, v_n^X\rangle\urcorner$ with variables v_i^X due to Remark 1. In case of the empty S-set $X = \mathbf{0}_S$ the only element in $T_\Sigma(Y)^{\mathbf{0}_S}$ is represented by the empty tuple $()$ *thus we have $\mathbf{syn}(\,()\,) = \ulcorner\langle\rangle\urcorner$.*

First-order expressions are strings of symbols. For convenience we drop, however, the delimiter signs $\ulcorner\ldots\urcorner$ in the following definition.

Definition 5 (Expressions: Syntax)**.** *For a signature $\Sigma = (S, P, F, ar, in, out)$, we define inductively and in parallel a family FE_Σ of sets $FE_\Sigma(X)$ of **(first-order) Σ-expressions** Ex **on** X, $X \rhd Ex$ in symbols, where X varies over all S-sets of variables, i.e., all elements in Var_S:*

1. *Atomic expressions:*
   (a) *Equation: $X \rhd (t_1 = t_2)$ for any $s \in S$ and any $t_1, t_2 \in T_\Sigma(X)_s$.*
   (b) *Relational Atom: $X \rhd \mathtt{p}\,\mathbf{syn}(\boldsymbol{t})$ for any $\mathtt{p} \in P$ and any S-map $\boldsymbol{t} : ar(\mathtt{p}) \to T_\Sigma(X)$.*
2. *Everything: $X \rhd \top\langle\rangle$ for any S-set X of variables.*
3. *Void: $X \rhd \bot\langle\rangle$ for any S-set X of variables.*
4. *Conjunction: $X \rhd (Ex_1 \wedge Ex_2)$ for any expressions $X \rhd Ex_1$ and $X \rhd Ex_2$.*
5. *Disjunction: $X \rhd (Ex_1 \vee Ex_2)$ for any expressions $X \rhd Ex_1$ and $X \rhd Ex_2$.*
6. *Implication: $X \rhd (Ex_1 \to Ex_2)$ for any expressions $X \rhd Ex_1$ and $X \rhd Ex_2$.*
7. *Negation: $X \rhd \neg Ex$ for any expression $X \rhd Ex$.*
8. *Quantification: $X \rhd \exists(\mathbf{syn}(Y \backslash X) : Ex)$ and $X \rhd \forall(\mathbf{syn}(Y \backslash X) : Ex)$ for any expression $Y \rhd Ex$ and any proper S-inclusion $X \subset Y$.*

Remark 5 (Syntax of expressions)**.** *There are three changes compared to the general Definition 8 in [1]. First, we do not use arbitrary S-maps between S-sets of variables but only proper inclusion S-maps $in_{X,Y} : X \hookrightarrow Y$*

in quantifications. Second, these inclusion S-maps are not considered to be morphisms in the category Var *of "variable declarations" for the institutions of statements defined in Section 5.1. Third, we adapt the traditional notation for quantification, i.e., we represent the proper inclusion S-maps* $in_{X,Y} : X \hookrightarrow Y$ *by the complements* $Y \setminus X$ *in* Set^S.

Attention, we do not keep track of the "free variables" that actually appear in an expression (compare Remark 3). $X \triangleright Ex$ *ensures, however, that all "free variables" in* Ex *are contained in* X*. So, in case of quantification, it may happen that none of the variables in* $Y \setminus X$ *appears as a "free variable" in* Ex*!*

Our design decision, to explicitly declare first all the variables $X \triangleright \ldots$*, which are available, before actually constructing expressions using only the variables in* X*, together with our requirement* $X \subset Y$ *for quantification has a subtle but important consequence. If* $X \triangleright Ex$ *then none of the variables in* X *appears as a "bound variable" in* Ex*. Other variables than the ones in* X *can appear, however, multiple times as "bound variables" in* Ex *but only in non-overlapping "sub-expressions". Lately, we learned that these syntactic restrictions correspond to* Barendregt's variable convention *for the* λ*-calculus (see [20], p. 26):*

- *bound variables are distinct from free variables, and*
- *all binders bind variables not already in scope.*

Every predicate symbol $\mathtt{p} \in P$ *is reborn as the* Σ*-expression* $ar(\mathtt{p}) \triangleright \mathtt{p}\,\mathbf{syn}(\eta_{ar(\mathtt{p})})$ *with* $\eta_{ar(\mathtt{p})} : ar(\mathtt{p}) \to T_\Sigma(ar(\mathtt{p}))$*. Keep in mind, that* $\mathbf{syn}(\eta_{ar(\mathtt{p})}) = \ulcorner\langle v_1^{ar(\mathtt{p})}, \ldots, v_n^{ar(\mathtt{p})}\rangle\urcorner$ *with* $n = |ar(\mathtt{p})|$ *according to Remark 4.*

Remark 6 (Everything and Void)**.** *We consider* $\top$ *and* $\bot$ *not as* logical constants *but as* built-in nullary predicate symbols*, i.e.,* $ar(\top) = ar(\bot) = \mathbf{0}_S$*. Therefore, we are not using symbols like* $\boldsymbol{T}$ *or* $\boldsymbol{F}$*, for example. Built-in, means, especially, that* $\top$ *and* $\bot$ *do have a fixed semantics for any* Σ*-structure* $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$*, namely* $\bot^{\mathcal{A}} = \emptyset$ *and* $\top^{\mathcal{A}} = A^{\mathbf{0}_S} = \{!_A : \mathbf{0}_S \to A\}$*. Be aware, that the string* $\ulcorner\langle\rangle\urcorner$ *in* $X \triangleright \top\langle\rangle$ *and* $X \triangleright \bot\langle\rangle$ *encodes the initial morphism* $!_{T_\Sigma(X)} : \mathbf{0}_S \to T_\Sigma(X)$*.*

Analogously, the equation symbol "=" can be seen to represent an S-indexed family of binary built-in predicate symbols $=_s$ *with a fixed semantics in any* Σ*-structure. Finally, built-in means that none of the symbols* $=, \bot, \top$ *appears in any of the sets* N_{sort}*,* N_{pred}*,* N_{fun} *or* N_{var}*.*

Remark 7 (Closed expressions: Syntax)**.** Σ*-expressions of the form* $\mathbf{0}_S \triangleright Ex$ *will be called* ***closed*** Σ***-expressions****. They represent nullary derived predicates that can be either valid or not valid in a given* Σ*-structure (compare Remark 6, Remark 9 and Subsection 4.2.5).*

Example 4 (Expressions: Syntax)**.** *We continue Example 3 and discuss some* Σ*-expressions for the sample signature in Example 1. For convenience, we will drop some parentheses and simplify the notations for encodings of S-sets of variables introduced in Remark 1.*

The following closed Σ*-expression, which we identify by the auxiliary name* `bfact`*,*

$$\mathbf{0}_S \triangleright \forall([x_1, x_2, x_3\colon prs] : \mathtt{par}\langle x_1, x_2, x_3\rangle \longrightarrow \mathtt{bef}\langle\mathtt{born}\langle x_2\rangle, \mathtt{born}\langle x_1\rangle\rangle \land \mathtt{bef}\langle\mathtt{born}\langle x_3\rangle, \mathtt{born}\langle x_1\rangle\rangle\,)$$

claims that biological parents are always born before their children.

Our main methodological point is, however, to consider expressions as syntactic representations of derived predicates *enabling us to denote properties in an anonymous way. We may say, for example, that the sample signature in Example 1 is redundant in the sense that the property, denoted by the basic predicate* `bef`*, could be also represented by the following* Σ*-expression*

$$\mathtt{nbf} \;:=\; (x_1, x_2\colon date) \triangleright \exists([y_1\colon nat] : \mathtt{plus}\langle x_1, y_1\rangle = x_2\,)$$

which we consider as a derived predicate with arity $(x_1, x_2\colon date)$ *and auxiliary name* `nbf`*. The following atomic* Σ*-expression gives us the property "born before" for two persons at hand*

$$(x_1, x_2\colon prs) \triangleright \mathtt{bef}\langle\mathtt{born}\langle x_1\rangle, \mathtt{born}\langle x_2\rangle\rangle\,.$$

The next Σ*-expression represents the property "being sibling of someone" thereby hiding the witnesses for a* *corresponding statement about a certain person*

$$\texttt{sbl} \ := \ (x_1 \colon prs) \triangleright \exists([y_1, y_2, y_3 \colon prs] : \neg(x_1 = y_1) \wedge \texttt{par}\langle x_1, y_2, y_3\rangle \wedge \texttt{par}\langle y_1, y_2, y_3\rangle).$$

Our last sample Σ-expression, with auxiliary name `hst`*, represents the bit more intrinsic property "being half-sister of someone"*

$$(x_1 \colon prs) \triangleright \texttt{fem}(x_1) \wedge \exists([y_1, y_2, y_3, y_4 \colon prs] : \neg(x_1 = y_1) \wedge \texttt{par}\langle x_1, y_2, y_3\rangle \\ \wedge ((\neg(y_3 = y_4) \wedge \texttt{par}\langle y_1, y_2, y_4\rangle) \vee (\neg(y_2 = y_4) \wedge \texttt{par}\langle y_1, y_4, y_3\rangle))).$$

3.3. Terms and First-Order Expressions: Semantics

As mentioned in Section 1, the semantics of a Σ-term in a given Σ-structure should be a derived operation. To define those operations, we consider the **evaluation of Σ-terms** in Σ-structures: Let a Σ-structure $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ be given. Based on Definition 4 and employing the fixed semantics $\mathrm{op}^{\mathcal{A}}$ of all operation symbols in F, we can inductively extend any assignment $\boldsymbol{a} \colon X \to A$ of an S-set X of variables in the carrier A to an S-map $\boldsymbol{a}^\circ : T_\Sigma(X) \to A$ such that

$$\eta_X ; \boldsymbol{a}^\circ = \boldsymbol{a}\,. \qquad (3)$$

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\boldsymbol{a}"'] & T_\Sigma(X) \arrow[d, "\boldsymbol{a}^\circ"] \\ & A \end{tikzcd}$$

This allows us to define the **semantics $t^{\mathcal{A}}$ of a Σ-term** $\ulcorner t \urcorner \in T_\Sigma(X)_s$, $s \in S$ in a Σ-structure $\mathcal{A}$ as the map $t^{\mathcal{A}} : A^X \to A_s$ defined by $t^{\mathcal{A}}(\boldsymbol{a}) := \boldsymbol{a}_s^\circ(t)$ for all $\boldsymbol{a} \colon X \to A$. We call $t^{\mathcal{A}}$ the derived operation in $\mathcal{A}$ represented by t. Each variable $\ulcorner x \urcorner \in T_\Sigma(X)_s$ represents a **projection** $x^{\mathcal{A}} : A^X \to A_s$ (compare Remark 2).

We can go even further: Any Σ-substitution $\boldsymbol{t} : Y \to T_\Sigma(X)$ defines a **derived operation $\boldsymbol{t}^{\mathcal{A}}$** : $A^X \to A^Y$ in $\mathcal{A}$ (in the opposite direction!) with

$$\boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}) := \boldsymbol{t}; \boldsymbol{a}^\circ \text{ , i.e., } (\boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}))_s(y) = (\boldsymbol{t}_s(y))^{\mathcal{A}}(\boldsymbol{a}) \quad \text{for all } \boldsymbol{a} \in A^X,\, s \in S,\, y \in Y_s. \qquad (4)$$

Since $A^{\mathbf{0}_S}$ is a singleton, the S-inclusion $!_{T_\Sigma(X)} : \mathbf{0}_S \hookrightarrow T_\Sigma(X)$ represents a constant operation $!^{\mathcal{A}}_{T_\Sigma(X)} : A^X \to A^{\mathbf{0}_S}$. Moreover, the canonical injective S-map $\eta_X \colon X \to T_\Sigma(X)$ represents the identity S-map on A^X:

$$\eta_X^{\mathcal{A}} = id_{A^X} : A^X \to A^X \quad \text{since} \quad \eta_X^\circ = id_{T_\Sigma(X)}. \qquad (5)$$

Example 5 (Terms: Semantics)**.** *We discuss the semantics of the five non-trivial Σ-terms from Example 3 in the Σ-structure $\mathcal{A}$ defined in Example 2.*

For the "derived constant" $\texttt{pl}\langle\texttt{sp}\langle\,\rangle, \texttt{one}\langle\,\rangle\rangle^{\mathcal{A}} : A^{\mathbf{0}_S} \to A_{date}$ *we have* $\texttt{pl}\langle\texttt{sp}\langle\,\rangle, \texttt{one}\langle\,\rangle\rangle^{\mathcal{A}}() = 20.10.1958$. $\texttt{pl}\langle x_1, \texttt{one}\langle\,\rangle\rangle^{\mathcal{A}} : A^{(x_1 \colon date)} \to A_{date}$ *computes for each date $d \in A_{date}$ the date of the next day.* $\texttt{pl}\langle\texttt{sp}\langle\,\rangle, x_1\rangle^{\mathcal{A}} : A^{(x_1 \colon nat)} \to A_{date}$ *provides for each natural number $n \in A_{nat}$ the date of the n-th day after 19.10.1958, while* $\texttt{pl}\langle\texttt{born}\langle x_1\rangle, x_2\rangle^{\mathcal{A}} : A^{(x_1 \colon prs, x_2 \colon nat)} \to A_{date}$ *computes for each person $p \in A_{prs}$ and each natural number $n \in A_{nat}$ the date of the n-th day after the day of birth of person p and* $\texttt{pl}\langle\texttt{born}\langle x_1\rangle, \texttt{one}\langle\,\rangle\rangle^{\mathcal{A}} : A^{(x_1 \colon prs)} \to A_{date}$ *provides the date of the next day after the day of birth of person p.*

Our idea is, to consider first-order Σ-expressions as derived predicates with a fixed semantics in any Σ-structure $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$. To define the semantics of a Σ-expression $X \triangleright Ex$ in $\mathcal{A}$, we first have to define the semantics $[\![\, X \,]\!]^{\mathcal{A}}$ of the S-set X of variables in $\mathcal{A}$. This is simply the set $[\![\, X \,]\!]^{\mathcal{A}} := A^X$ of all assignments of X in A. A Σ-expression Ex is built upon Σ-terms, predicate symbols and substitutions. The semantics of Σ-terms and substitutions in $\mathcal{A}$ we do have already at hand, thus it remains to define the semantics $[\![\, Ex \,]\!]^{\mathcal{A}}_X$ of Σ-expressions $X \triangleright Ex$ in $\mathcal{A}$ employing the fixed semantics $\mathrm{p}^{\mathcal{A}}$ of all predicate symbols in P. Relying on Definition 5, we will inductively define the family of sets $[\![\, Ex \,]\!]^{\mathcal{A}}_X \subseteq [\![\, X \,]\!]^{\mathcal{A}}$ of **valid assignments (solutions)** of $X \triangleright Ex$ in $\mathcal{A}$. For convenience, the interested reader may use instead of $\boldsymbol{a} \in [\![\, Ex \,]\!]^{\mathcal{A}}_X$, the more traditional notation $\boldsymbol{a} \models^{\Sigma}_{\mathcal{A}} X \triangleright Ex$ to follow the definitions.

Given an S-inclusion $X \subseteq Y$, we say that an assignment $\boldsymbol{b} : Y \to A$ is an **expansion** of an assignment $\boldsymbol{a} \colon X \to A$ to Y if, and only if, $in_{X,Y} ; \boldsymbol{b} = \boldsymbol{a}$.

$$\begin{tikzcd}
X \arrow[rr, hook, "in_{X,Y}"] \arrow[dr, "\boldsymbol{a}"'] & = & Y \arrow[dl, "\boldsymbol{b}"] \\
& A &
\end{tikzcd}$$

Definition 6 (Expressions: Semantics). *The semantics of Σ-expressions in an arbitrary, but fixed, Σ-structure $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ is defined inductively:*

1. *Atomic expressions:*

   (a) *Equation: $\boldsymbol{a} \in \llbracket\, t_1 = t_2 \,\rrbracket^{\mathcal{A}}_X$ iff $t_1^{\mathcal{A}}(\boldsymbol{a}) = t_2^{\mathcal{A}}(\boldsymbol{a})$.*
   (b) *Relational Atom: $\boldsymbol{a} \in \llbracket\, \mathtt{p}\, \mathbf{syn}(\boldsymbol{t}) \,\rrbracket^{\mathcal{A}}_X$ iff $\boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}) \in \mathtt{p}^{\mathcal{A}}$.*
2. *Everything: $\llbracket\, \top\langle\rangle \,\rrbracket^{\mathcal{A}}_X := \llbracket\, X \,\rrbracket^{\mathcal{A}} = A^X$*
3. *Void: $\llbracket\, \bot\langle\rangle \,\rrbracket^{\mathcal{A}}_X := \emptyset$*
4. *Conjunction: $\llbracket\, (Ex_1 \land Ex_2) \,\rrbracket^{\mathcal{A}}_X := \llbracket\, Ex_1 \,\rrbracket^{\mathcal{A}}_X \cap \llbracket\, Ex_2 \,\rrbracket^{\mathcal{A}}_X$*
5. *Disjunction: $\llbracket\, (Ex_1 \lor Ex_2) \,\rrbracket^{\mathcal{A}}_X := \llbracket\, Ex_1 \,\rrbracket^{\mathcal{A}}_X \cup \llbracket\, Ex_2 \,\rrbracket^{\mathcal{A}}_X$*
6. *Implication: $\boldsymbol{a} \in \llbracket\, Ex_1 \to Ex_2 \,\rrbracket^{\mathcal{A}}_X$ iff $\boldsymbol{a} \in \llbracket\, Ex_2 \,\rrbracket^{\mathcal{A}}_X$ whenever $\boldsymbol{a} \in \llbracket\, Ex_1 \,\rrbracket^{\mathcal{A}}_X$*
7. *Negation: $\llbracket\, \neg Ex \,\rrbracket^{\mathcal{A}}_X := A^X \setminus \llbracket\, Ex \,\rrbracket^{\mathcal{A}}_X$*
8. *Existential quantification: $\boldsymbol{a} \in \llbracket\, \exists(\mathbf{syn}(Y \backslash X) : Ex) \,\rrbracket^{\mathcal{A}}_X$ iff there exists an expansion $\boldsymbol{b} \colon Y \to A$ of $\boldsymbol{a} \colon X \to A$ to Y such that $\boldsymbol{b} \in \llbracket\, Ex \,\rrbracket^{\mathcal{A}}_Y$.*

   Universal quantification: $\boldsymbol{a} \in \llbracket\, \forall(\mathbf{syn}(Y \backslash X) : Ex) \,\rrbracket^{\mathcal{A}}_X$ iff for all expansions $\boldsymbol{b} \colon Y \to A$ of $\boldsymbol{a} \colon X \to A$ to Y we have $\boldsymbol{b} \in \llbracket\, Ex \,\rrbracket^{\mathcal{A}}_Y$.

Remark 8 (Expressions: Semantics). *As mentioned in Remark 5, every predicate symbol $\mathtt{p} \in P$ is reborn as the Σ-expression $ar(\mathtt{p}) \rhd \mathtt{p}\, \mathbf{syn}(\eta_{ar(\mathtt{p})})$ with $\eta_{ar(\mathtt{p})} : ar(\mathtt{p}) \to T_\Sigma(ar(\mathtt{p}))$. Definition 6 and Equation (5) ensure that also their semantics coincides $\llbracket\, \mathtt{p}\, \mathbf{syn}(\eta_{ar(\mathtt{p})}) \,\rrbracket^{\mathcal{A}}_{ar(\mathtt{p})} = \mathtt{p}^{\mathcal{A}}$.*

The universal quantification $\forall(\mathbf{syn}(Y \backslash X) : Ex)$ is trivially valid for $\boldsymbol{a}$ if there is no expansion of $\boldsymbol{a}$ to Y at all, while the existential quantification $\exists(\mathbf{syn}(Y \backslash X) : Ex)$ is not valid, in this case.

*Two expressions $X \rhd Ex_1$ and $X \rhd Ex_2$ are **semantical equivalent**, $X \rhd Ex_1 \equiv Ex_2$ in symbols, if, and only if, $\llbracket\, Ex_2 \,\rrbracket^{\mathcal{A}}_X = \llbracket\, Ex_2 \,\rrbracket^{\mathcal{A}}_X$ for all Σ-structures $\mathcal{A}$. Definition 6 ensures that we do have the usual semantic equivalences available. In particular, conjunction and disjunction are associative; thus we can drop, for convenience, the corresponding parenthesis as we have done it already at some places in Example 4.*

Remark 9 (Closed expressions: Semantics). *We consider a closed expression $\mathbf{0}_S \rhd Ex$ (see Remark 7). $\llbracket\, \mathbf{0}_S \,\rrbracket^{\mathcal{A}} = A^{\mathbf{0}_S}$ is a singleton with the initial morphism $!_A : \mathbf{0}_S \to A$ as the only element. In such a way, we have either $\llbracket\, Ex \,\rrbracket^{\mathcal{A}}_{\mathbf{0}_S} = \llbracket\, \top\langle\rangle \,\rrbracket^{\mathcal{A}}_{\mathbf{0}_S} = \{!_A\}$, i.e., $!_A \models^{\Sigma}_{\mathcal{A}} \mathbf{0}_S \rhd Ex$, or $\llbracket\, Ex \,\rrbracket^{\mathcal{A}}_{\mathbf{0}_S} = \llbracket\, \bot\langle\rangle \,\rrbracket^{\mathcal{A}}_{\mathbf{0}_S} = \emptyset$, i.e., $!_A \not\models^{\mathcal{A}}_{\Sigma} \mathbf{0}_S \rhd Ex$.*

Example 6 (Expressions: Semantics). *We consider the sample Σ-structure $\mathcal{A}$ in Example 2 and the derived predicates in Example 4.*

For the closed Σ-expression $\mathtt{bfact}$ we do have $\llbracket\, \mathtt{bfact} \,\rrbracket^{\mathcal{A}}_{\mathbf{0}_S} = \{!_A\}$, i.e., $!_A \models^{\Sigma}_{\mathcal{A}} \mathbf{0}_S \rhd \mathtt{bfact}$. Note, that this is only the case since $\mathtt{par}^{\mathcal{A}} \subseteq A^{ar(\mathtt{par})}$ is defined in such a way that $(p_1, p_2, p_3) \in \mathtt{par}^{\mathcal{A}}$ means that p_1 is the child of p_2 and p_3.

The semantics $\llbracket\, \mathtt{nbf} \,\rrbracket^{\mathcal{A}}_{(x_1, x_2 \colon date)}$ of the derived predicate $\mathtt{nbf}$ in $\mathcal{A}$ coincides indeed with the semantics $\mathtt{bef}^{\mathcal{A}}$ of the basic predicate $\mathtt{bef}$ in $\mathcal{A}$. $\llbracket\, \mathtt{sbl} \,\rrbracket^{\mathcal{A}}_{(x_1 \colon prs)}$ gives us all the knowledge about "being a sibling of someone", that can be extracted from the information in $\mathcal{A}$, and $\llbracket\, \mathtt{hst} \,\rrbracket^{\mathcal{A}}_{(x_1 \colon prs)}$ the corresponding knowledge about "being a half-brother of someone". Remember, that there is no information in $\mathtt{par}^{\mathcal{A}}$ about the parents of a person, if one of the parents is born before 01.01.1900 or outside Europe.

4. Change of Base

In the last section we introduced and discussed basic concepts, structures and constructions connected to a single many-sorted first-order signature. In this section, we investigate how these concepts, structures and constructions are related across different signatures.

4.1. Maps Between Sets of Sort Symbols

By Sort we denote the full subcategory of Set with objects all finite subsets of N_{sort} and morphisms all maps between those finite sets of sort symbols.

According to Section 2, we do have for every $S \in |\mathsf{Sort}|$ the category Set^S of S-sets, its subcategory Set^S_d of disjoint S-sets and the category Set/S of S-typed sets at hand.

For any map $\varphi : S \to S'$ we obtain a (reduct) functor $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ defined by $\mathbf{R}_\varphi(B') := (B'_{\varphi(s)} \mid s \in S)$ for all S'-sets $B' = (B'_{s'} \mid s' \in S')$ and $\mathbf{R}_\varphi(f') := (f'_{\varphi(s)} \mid s \in S)$ for all S'-maps $f' = (f'_{s'} : A'_{s'} \to B'_{s'} \mid s' \in S') : A' \to B'$. We can consider the sets S, S' as discrete categories, thus $\mathbf{R}_\varphi$ is simply defined by precomposing the functors $A', B' : S' \to \mathsf{Set}$ and the natural transformation $f' : A' \to B'$ with the functor φ. In such a way, the functoriality of $\mathbf{R}_\varphi$ is ensured by the right equation in (2). In addition, the left equation in (1) entails that the assignments $\varphi \mapsto \mathbf{R}_\varphi$ establish a functor $\mathbf{R} : \mathsf{Sort}^{op} \to \mathsf{CAT}$.

Any fixed choice of finite sums in Set allows us to define a functor $\mathbf{S}_\varphi : \mathsf{Set}^S \to \mathsf{Set}^{S'}$ with $\mathbf{S}_\varphi(A) := \{\coprod\{A_s \mid \varphi(s) = s'\} \mid s' \in S'\}$ for all S-sets A and

$\mathbf{S}_\varphi(f) := \{\coprod\{f_s \mid \varphi(s) = s'\} : \coprod\{A_s \mid \varphi(s) = s'\} \to \coprod\{B_s \mid \varphi(s) = s'\} \mid s' \in S'\}$

for all S-maps $f = (f_s : A_s \to B_s \mid s \in S) : A \to B$. $\mathbf{S}_\varphi$ becomes left adjoint to $\mathbf{R}_\varphi$.

The crucial technical trick is, that we can choose the sums in Set to be the union for finite families of disjoint sets, thus we have $\mathbf{S}_\varphi(A)_{s'} := \bigcup\{A_s \mid \varphi(s) = s'\}$ for all disjoint S-sets A and all $s' \in S'$. Keep in mind that $\mathbf{S}_\varphi(A)_{s'} = \emptyset$ if there is no $s \in S$ with $\varphi(s) = s'$, i.e., if $s' \in S' \setminus \varphi(S)$. This "union trick" ensures two essential technical prerequisites:

1. The functor $\mathbf{S}_\varphi : \mathsf{Set}^S \to \mathsf{Set}^{S'}$ restricts to a functor $\mathbf{S}^d_\varphi : \mathsf{Set}^S_d \to \mathsf{Set}^{S'}_d$ (mimicking the functor $(\_;\varphi) : \mathsf{Set}/S \to \mathsf{Set}/S'$).
2. For families of non-disjoint sets the chosen finite sums will be, in general, not associative on the nose, thus the assignments $S \mapsto \mathsf{Set}^S$ and $\varphi \mapsto \mathbf{S}_\varphi$ do not define a functor but only a *pseudo functor* from Sort into CAT.
3. Union of sets is, however, associative on the nose, thus the assignments $S \mapsto \mathsf{Set}^S_d$ and $\varphi \mapsto \mathbf{S}^d_\varphi$ define a functor $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$.

Note, $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ is a "proper reduct functor" only for injective φ and restricts then also to a functor from $\mathsf{Set}^{S'}_d$ to Set^S_d. For non-injective $\varphi : S \to S'$ the functor $\mathbf{R}_\varphi$ produces, however, identical copies of some component sets and component maps, respectively, and restricts therefore not to a functor from $\mathsf{Set}^{S'}_d$ to Set^S_d.

The functors $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ are later utilized to translate structures while the functors $\mathbf{S}^d_\varphi : \mathsf{Set}^S_d \to \mathsf{Set}^{S'}_d$ will enable us to translate syntactic entities, like terms and expressions, in the opposite direction. The adjunctions $\mathbf{S}_\varphi \dashv \mathbf{R}_\varphi : \mathsf{Set}^S \to \mathsf{Set}^{S'}$ pave the way for proving satisfaction conditions. To facilitate these proofs, we use Lawvere's characterization of adjunctions by means of comma categories [21,22].

$\mathbf{S}_\varphi : \mathsf{Set}^S \to \mathsf{Set}^{S'}$ left adjoint to $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ means nothing but that there exists an isomorphism $\mathbf{B}_\varphi : (\mathbf{S}_\varphi \downarrow \mathsf{Set}^{S'}) \to (\mathsf{Set}^S \downarrow \mathbf{R}_\varphi)$ between comma categories such that the following triangle of functors commutes.

$$\begin{tikzcd}
(\mathbf{S}_\varphi \downarrow \mathsf{Set}^{S'}) \arrow[rr, "\mathbf{B}_\varphi"] \arrow[dr, "\Pi_L"'] & = & (\mathsf{Set}^S \downarrow \mathbf{R}_\varphi) \arrow[dl, "\Pi_R"] \\
& \mathsf{Set}^S \times \mathsf{Set}^{S'} &
\end{tikzcd}$$

More precisely, we will need a one-to-one correspondence between S-sorted variable assignments and S'-sorted variable assigments. Families of sets of variables are disjoint while carriers of structures are not. Therefore, we will later need the following restriction of $\mathbf{B}_\varphi$ to subcategories of $(\mathbf{S}_\varphi \downarrow \mathsf{Set}^{S'})$ and $(\mathsf{Set}^S \downarrow \mathbf{R}_\varphi)$, constructed by means of the inclusion functors $\mathbf{E}_S : \mathsf{Set}^S_d \hookrightarrow \mathsf{Set}^S$ and $\mathbf{E}_{S'} : \mathsf{Set}^{S'}_d \hookrightarrow \mathsf{Set}^{S'}$.

$$\begin{tikzcd}
(\mathbf{E}_S;\mathbf{S}_\varphi \downarrow \mathsf{Set}^{S'}) = (\mathbf{S}^d_\varphi;\mathbf{E}'_S \downarrow \mathsf{Set}^{S'}) \arrow[rr, "\mathbf{B}_\varphi"] \arrow[dr, "\Pi_L"'] & \arrow[d, phantom, "="] & (\mathbf{E}_S \downarrow \mathbf{R}_\varphi) \arrow[dl, "\Pi_R"] \\
& \mathsf{Set}^S_d \times \mathsf{Set}^{S'} &
\end{tikzcd} \tag{6}$$

$$\begin{tikzcd}
\mathsf{Set}^{S'}_d \arrow[d, hook, "\mathbf{E}'_S"'] & \mathsf{Set}^S_d \arrow[l, "\mathbf{S}^d_\varphi"'] \arrow[d, hook, "\mathbf{E}_S"'] \\
\mathsf{Set}^{S'} \arrow[r, shift right, "\mathbf{R}_\varphi"'] & \mathsf{Set}^S \arrow[l, shift right, "\mathbf{S}_\varphi"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{S}^d_\varphi(X) \arrow[d, "\boldsymbol{a}'"', ""{name=A}] & X \arrow[l, mapsto, "\mathbf{S}^d_\varphi"'] \arrow[d, "\mathbf{B}_\varphi(\boldsymbol{a}')", ""'{name=B}] \\
A' \arrow[r, mapsto, "\mathbf{R}_\varphi"] & \mathbf{R}_\varphi(A')
\arrow[from=A, to=B, mapsto, "\mathbf{B}_\varphi"]
\end{tikzcd}$$

The objects in $(\mathbf{S}^d_\varphi;\mathbf{E}'_S \downarrow \mathsf{Set}^{S'})$ are all triples $(X, \boldsymbol{a}', A')$ of a disjoint S-set X, an S'-set A' and an S'-map $\boldsymbol{a}' : \mathbf{S}^d_\varphi(X) \to A'$. The morphisms $(\mu, f') : (X, \boldsymbol{a}', A') \to (Y, \boldsymbol{b}', B')$ in $(\mathbf{S}^d_\varphi;\mathbf{E}'_S \downarrow \mathsf{Set}^{S'})$ are given by an S-map $\mu : X \to Y$ and an S'-map $f' : A' \to B'$ such that $\boldsymbol{a}'; f' = \mathbf{S}^d_\varphi(\mu); \boldsymbol{b}'$ (see the left diagram below). Π_L is the obvious projection.

$$\begin{tikzcd}
X \arrow[d, "\mu"'] & \mathbf{S}^d_\varphi(X) \arrow[r, "\boldsymbol{a}'"] \arrow[d, "\mathbf{S}^d_\varphi(\mu)"'] \arrow[dr, phantom, "="] & A' \arrow[d, "f'"] & A' \arrow[d, "f'"] \\
Y & \mathbf{S}^d_\varphi(Y) \arrow[r, "\boldsymbol{b}'"] & B' & B'
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[d, "\mu"'] & X \arrow[r, "\boldsymbol{a}"] \arrow[d, "\mu"'] \arrow[dr, phantom, "="] & \mathbf{R}_\varphi(A') \arrow[d, "\mathbf{R}_\varphi(f')"] & A' \arrow[d, "f'"] \\
Y & Y \arrow[r, "\boldsymbol{b}"] & \mathbf{R}_\varphi(B') & B'
\end{tikzcd}$$

The objects in $(\mathbf{E}_S \downarrow \mathbf{R}_\varphi)$ are all triples $(X, \boldsymbol{a}, A')$ of a disjoint S-set X, an S'-set A' and an S-map $\boldsymbol{a} : X \to \mathbf{R}_\varphi(A')$. The morphisms $(\mu, f') : (X, \boldsymbol{a}, A') \to (Y, \boldsymbol{b}, B')$ in $(\mathbf{E}_S \downarrow \mathbf{R}_\varphi)$ are given by an S-map $\mu : X \to Y$ and an S'-map $f' : A' \to B'$ such that $\boldsymbol{a}; \mathbf{R}_\varphi(f') = \mu; \boldsymbol{b}$ (see the right diagram above). Π_R is the obvious projection.

The commutativity of the triangle of functors in (6) ensures that the isomorphism $\mathbf{B}_\varphi$ restricts to a bijective map

$$\mathbf{B}_\varphi : A'^{\mathbf{S}^d_\varphi(X)} \to \mathbf{R}_\varphi(A')^X \quad \text{for all disjoint } S\text{-sets } X \text{ and all } S'\text{-sets } A'. \tag{7}$$

These canonical bijections appear, at least implicitly, in all proofs of the satisfaction condition for (fragments of) first-order logic (see [17], for example) and are called "corresponding assignments" in [23]. Moreover, we get

$$\mathbf{B}_\varphi; (\_; \mathbf{R}_\varphi(f')) = (\_; f'); \mathbf{B}_\varphi \quad \text{for any disjoint } S\text{-set } X \text{ and any } S'\text{-map } f' : A' \to B'. \tag{8}$$

$$\begin{tikzcd}
A'^{\mathbf{S}^d_\varphi(X)} \arrow[r, "\mathbf{B}_\varphi"] \arrow[d, "(\_;f')"'] \arrow[dr, phantom, "="] & \mathbf{R}_\varphi(A')^X \arrow[d, "(\_;\mathbf{R}_\varphi(f'))"] \\
B'^{\mathbf{S}^d_\varphi(X)} \arrow[r, "\mathbf{B}_\varphi"] & \mathbf{R}_\varphi(B')^X
\end{tikzcd}
\qquad
\begin{tikzcd}
A'^{\mathbf{S}^d_\varphi(X)} \arrow[d, "(\_;f')"'] \arrow[dr, phantom, "="] & \mathbf{R}_\varphi(A')^X \arrow[l, "\mathbf{B}^{-1}_\varphi"'] \arrow[d, "(\_;\mathbf{R}_\varphi(f'))"] \\
B'^{\mathbf{S}^d_\varphi(X)} & \mathbf{R}_\varphi(B')^X \arrow[l, "\mathbf{B}^{-1}_\varphi"']
\end{tikzcd}$$

This follows immediately from the commutativity in (6) and the functoriality of $\mathbf{S}^d_\varphi$:

$$\begin{tikzcd}
X \arrow[d, "id_X"'] & \mathbf{S}^d_\varphi(X) \arrow[rr, "a'"] \arrow[d, "\mathbf{S}^d_\varphi(id_X)=id_{\mathbf{S}^d_\varphi(X)}" description] \arrow[drr, phantom, "="] & & A' \arrow[d, "f'"] \\
X & \mathbf{S}^d_\varphi(X) \arrow[rr, "a';f'"] & & B'
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[rr, "\mathbf{B}_\varphi(a')"] \arrow[d, "id_X"'] \arrow[drr, phantom, "="] & & \mathbf{R}_\varphi(A') \arrow[d, "\mathbf{R}_\varphi(f')"] & A' \arrow[d, "f'"] \\
X \arrow[rr, "\mathbf{B}_\varphi(a';f')"] & & \mathbf{R}_\varphi(B') & B'
\end{tikzcd}$$

Since the $\mathbf{B}_\varphi$ are isomorphisms, we also do have the equivalent statement:

$$(\_;\mathbf{R}_\varphi(f'));\mathbf{B}_\varphi^{-1} = \mathbf{B}_\varphi^{-1};(\_;f') \text{ for any disjoint } S\text{-set } X \text{ and any } S'\text{-map } f' : A' \to B'. \tag{9}$$

Symmetrically, it follows from the commutativity in (6) and the functoriality of $\mathbf{R}_\varphi$:

$$\mathbf{B}_\varphi;(\mu;\_) = (\mathbf{S}^d_\varphi(\mu);\_);\mathbf{B}_\varphi \quad \text{for any disjoint } S\text{-map } \mu : X \to Y \text{ and any } S'\text{-set } A'. \tag{10}$$

$$\begin{tikzcd}
X \arrow[d, "\mu"'] & \mathbf{S}^d_\varphi(X) \arrow[rr, "\mathbf{S}^d_\varphi(\mu);b'"] \arrow[d, "\mathbf{S}^d_\varphi(\mu)"'] \arrow[drr, phantom, "="] & & A' \arrow[d, "id_{A'}"] \\
Y & \mathbf{S}^d_\varphi(Y) \arrow[rr, "b'"] & & A'
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[rr, "\mathbf{B}_\varphi(\mathbf{S}^d_\varphi(\mu);b')"] \arrow[d, "\mu"'] \arrow[drr, phantom, "="] & & \mathbf{R}_\varphi(A') \arrow[d, "\mathbf{R}_\varphi(id_{A'})=id_{\mathbf{R}_\varphi(A')}" description] & A' \arrow[d, "id_{A'}"] \\
Y \arrow[rr, "\mathbf{B}_\varphi(b')"] & & \mathbf{R}_\varphi(A') & A'
\end{tikzcd}$$

Remark 10 (Pseudo Functors). *The original concept of* institution *[16,17] is based on the optimistic prerequisite that as well the reduction of models as the translation of syntax is functorial on the nose. As indicated in this section, we are able to meet this prerequisite, in case of first-order logic, if we work with S-indexed sets.*

It is maybe worth to mention that we could not meet this prerequisite if we utilize S-typed sets instead. A functorial translation of syntax we would get for free by simple post-composition, i.e., by the functors $(\_;\varphi) : \mathsf{Set}/S \to \mathsf{Set}/S'$. *And, for any fixed choice of pullbacks in* Set *we would get, for all* φ, *a functor from* Set/S' *to* Set/S *right adjoint to* $(\_;\varphi)$. *However, for any fixed choice of pullbacks in* Set *the composition of chosen pullbacks will be not functorial on the nose thus reduction of models would as well not be functorial but only pseudo-functorial!*

We have been facing this problem in [24] and guess that there are many situations where we are able to ensure functoriality on the nose for either "reduction of models" or "translation of syntax" while the other mechanism becomes pseudo-functorial only.

While indexed functors correspond to split (op)fibrations, indexed pseudo functors are just equivalent to (op)fibrations. It is maybe worth, to develop and explore, one day, a fibred counterpart of institutions simply based on (op)fibrations, to handle the obstacle of pseudo functors? One could, for example, relax the concept of split fibred frames *in [25] to a concept of* fibred frames. *For such a project we have to bear in mind that there are actually four possibilities to transform an indexed category into a split (op)fibration not only the standard Grothendieck construction usually presented in the literature [8].*

4.2. Signature Morphisms

In this subsection we introduce signature morphisms and define as well the reduction of structures as the translation of syntactic entities induced by signature morphisms.

Definition 7 (Signature Morphisms). *A* ***signature morphism*** $\varphi = (\varphi_{st}, \varphi_{rl}, \varphi_{fn}) : \Sigma \to \Sigma'$ *between signatures* $\Sigma = (S, P, F, ar, in, out)$ *and* $\Sigma' = (S', P', F', ar', in', out')$ *consists of*

- *a map* $\varphi_{st} : S \to S'$,
- *a map* $\varphi_{rl} : P \to P'$ *such that* $ar'(\varphi_{rl}(\mathtt{p})) = \mathbf{S}^d_{\varphi_{st}}(ar(\mathtt{p}))$ *for all* $\mathtt{p} \in P$, *and*
- *a map* $\varphi_{fn} : F \to F'$ *such that* $in'(\varphi_{fn}(\mathtt{op})) = \mathbf{S}^d_{\varphi_{st}}(in(\mathtt{op}))$ *and* $out'(\varphi_{fn}(\mathtt{op})) = \mathbf{S}^d_{\varphi_{st}}(out(\mathtt{op}))$ *for all* $\mathtt{op} \in F$.

The composition $\varphi;\psi : \Sigma \to \Sigma''$ of signature morphisms $\varphi : \Sigma \to \Sigma'$ and $\psi : \Sigma' \to \Sigma''$ is defined component-wise:

$$(\varphi;\psi)_{st} := \varphi_{st};\psi_{st}, \ (\varphi;\psi)_{rl} := \varphi_{rl};\psi_{rl} \quad \text{and} \quad (\varphi;\psi)_{fn} := \varphi_{fn};\psi_{fn}. \tag{11}$$

The functoriality of the assignments $\varphi_{st} \mapsto \mathbf{S}^d_{\varphi_{st}}$ ensures that $\varphi;\psi$ becomes indeed a signature morphism $\varphi;\psi : \Sigma \to \Sigma''$.

Sig denotes the category of all many-sorted signatures and signature morphisms. By construction, we do have the obvious projection functor $\Pi_{st} : \mathsf{Sig} \to \mathsf{Sort}$ which is a split opfibration.

4.2.1. Reducts

Signature morphisms are defined in a way that we can construct corresponding reducts of structures by means of the reduct functors $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ in Section 4.1 and the bijections in (7).

Definition 8 (Structure Reduct)**.** *Given a signature morphism $\varphi : \Sigma \to \Sigma'$ the **φ-reduct** of a Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$ is a Σ-structure $\mathcal{A}'\!\upharpoonright_\varphi = (A'\!\upharpoonright_\varphi, P^{\mathcal{A}'\upharpoonright_\varphi}, F^{\mathcal{A}'\upharpoonright_\varphi})$ with*

- *carrier* $A'\!\upharpoonright_\varphi := \mathbf{R}_{\varphi_{st}}(A') = (A'_{\varphi_{st}(s)} \mid s \in S)$,
- $\mathtt{p}^{\mathcal{A}'\upharpoonright_\varphi} := \mathbf{B}_{\varphi_{st}}(\varphi_{rl}(\mathtt{p})^{\mathcal{A}'}) \subseteq \mathbf{R}_{\varphi_{st}}(A')^{ar(\mathtt{p})}$ *for all predicate symbols* $\mathtt{p} \in P$*, and*

$$\begin{tikzcd}
\varphi_{rl}(\mathtt{p})^{\mathcal{A}'} \arrow[r, phantom, "\subseteq"] \arrow[d, "\mathbf{B}_{\varphi_{st}}"'] & A'^{\,ar'(\varphi_{rl}(\mathtt{p}))} = A'^{\,\mathbf{S}^d_{\varphi_{st}}(ar(\mathtt{p}))} \arrow[d, "\mathbf{B}_{\varphi_{st}}"] \\
\mathtt{p}^{\mathcal{A}'\upharpoonright_\varphi} \arrow[r, phantom, "\subseteq"] & \mathbf{R}_{\varphi_{st}}(A')^{ar(\mathtt{p})}
\end{tikzcd}$$

- $\mathtt{op}^{\mathcal{A}'\upharpoonright_\varphi} := \mathbf{B}^{-1}_{\varphi_{st}};\varphi_{fn}(\mathtt{op})^{\mathcal{A}'};\mathbf{B}_{\varphi_{st}}$ *for all operation symbols* $\mathtt{op} \in F$.

$$\begin{tikzcd}
A'^{\,in'(\varphi_{fn}(\mathtt{op}))} = A'^{\,\mathbf{S}^d_{\varphi_{st}}(in(\mathtt{op}))} \arrow[r, "\varphi_{fn}(\mathtt{op})^{\mathcal{A}'}"] & A'^{\,out'(\varphi_{fn}(\mathtt{op}))} = A'^{\,\mathbf{S}^d_{\varphi_{st}}(out(\mathtt{op}))} \arrow[d, "\mathbf{B}_{\varphi_{st}}"] \\
\mathbf{R}_{\varphi_{st}}(A')^{in(\mathtt{op})} \arrow[u, "\mathbf{B}^{-1}_{\varphi_{st}}"] \arrow[r, dashed, "\mathtt{op}^{\mathcal{A}'\upharpoonright_\varphi}"] & \mathbf{R}_{\varphi_{st}}(A')^{out(\mathtt{p})}
\end{tikzcd}$$

For later we should keep in mind that $\mathbf{B}_{\varphi_{st}} : \varphi_{rl}(\mathtt{p})^{\mathcal{A}'} \to \mathtt{p}^{\mathcal{A}'\upharpoonright_\varphi}$ is a bijection by construction and that $\mathbf{B}_{\varphi_{st}};\mathtt{op}^{\mathcal{A}'\upharpoonright_\varphi} = \varphi_{fn}(\mathtt{op})^{\mathcal{A}'};\mathbf{B}_{\varphi_{st}}$ by definition of $\mathtt{op}^{\mathcal{A}'\upharpoonright_\varphi}$.

Reducts of homomorphisms are also given by the functors $\mathbf{R}_{\varphi_{st}} : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ in Section 4.1, i.e., by reducts of S'-maps, while the homomorphism property of the reducts is ensured by the equations in (8) and (9).

Definition 9 (Homomorphism Reduct)**.** *Given a signature morphism $\varphi : \Sigma \to \Sigma'$ the **φ-reduct** of a Σ'-homomorphism $\varsigma' : \mathcal{A}' \to \mathcal{B}'$ between two Σ'-structures $\mathcal{A}'$ and $\mathcal{B}'$ is the Σ-homomorphism $\varsigma'\!\upharpoonright_\varphi : \mathcal{A}'\!\upharpoonright_\varphi \to \mathcal{B}'\!\upharpoonright_\varphi$ given by the S-map* $\varsigma'\!\upharpoonright_\varphi := \mathbf{R}_{\varphi_{st}}(\varsigma') = (\varsigma'_{\varphi_{st}(s)} \mid s \in S)$.

Notation: As one can see, the subscript notations φ_{st}, φ_{rl} and φ_{fn} can become quite heavy. When there is no danger of ambiguity, we may just drop the subscripts from now on.

For any signature morphism $\varphi : \Sigma \to \Sigma'$ the functoriality of $\mathbf{R}_\varphi : \mathsf{Set}^{S'} \to \mathsf{Set}^S$ entails that the assignments $\mathcal{A}' \mapsto \mathcal{A}'\!\upharpoonright_\varphi$ and $(\varsigma' : \mathcal{A}' \to \mathcal{B}') \mapsto (\varsigma'\!\upharpoonright_\varphi : \mathcal{A}'\!\upharpoonright_\varphi \to \mathcal{B}'\!\upharpoonright_\varphi)$ define a functor $\mathbf{Str}(\varphi) = (\_\!\upharpoonright_\varphi) : \mathsf{Str}(\Sigma') \to \mathsf{Str}(\Sigma)$.

On the global level the functoriality of $\mathbf{R} : \mathsf{Sort}^{op} \to \mathsf{CAT}$ entails for any signature morphisms $\varphi : \Sigma \to \Sigma'$ and $\psi : \Sigma' \to \Sigma''$ that $(\mathcal{A}''\!\upharpoonright_\psi)\!\upharpoonright_\varphi = \mathcal{A}''\!\upharpoonright_{\varphi;\psi}$ for all Σ''-structures $\mathcal{A}''$ and $(\varsigma''\!\upharpoonright_\psi)\!\upharpoonright_\varphi = \varsigma''\!\upharpoonright_{\varphi;\psi}$ for all Σ''-homomorphisms $\varsigma'' : \mathcal{A}'' \to \mathcal{B}''$. The fact that adjunctions compose ensures, in such a way, that the functor $\mathbf{R} : \mathsf{Sort}^{op} \to \mathsf{CAT}$ lifts up to a functor $\mathbf{Str} : \mathsf{Sig}^{op} \to \mathsf{CAT}$ given by the assignments $\Sigma \mapsto \mathsf{Str}(\Sigma)$ and $\varphi \mapsto \mathbf{Str}(\varphi)$.

4.2.2. Translation of Terms

For each signature $\Sigma = (S, P, F, ar, in, out)$ the assignments $X \mapsto T_\Sigma(X)$ define a map from Var_S into $|\mathsf{Set}_d^S|$. For convenience we can consider Var_S as a discrete subcategory of Set_d^S and use the functor notation $T_\Sigma : Var_S \to \mathsf{Set}_d^S$ for this map.

For any signature morphism $\varphi : \Sigma \to \Sigma'$ the corresponding translation of terms is given by a natural transformation $\overline{\varphi} : T_\Sigma \,;\mathbf{S}_\varphi^d \Rightarrow \mathbf{S}_\varphi^d \,; T_{\Sigma'} : Var_S \to \mathsf{Set}_d^{S'}$.

$$\begin{tikzcd}
Var_S \arrow[r, "T_\Sigma"] \arrow[d, "\mathbf{S}_\varphi^d"'] & \mathsf{Set}_d^S \arrow[d, "\mathbf{S}_\varphi^d"] \arrow[dl, Rightarrow, "\overline{\varphi}"'] \\
Var_{S'} \arrow[r, "T_{\Sigma'}"] & \mathsf{Set}_d^{S'}
\end{tikzcd}$$

For each S-set X of variables in Var_S the S'-map $\overline{\varphi}_X : \mathbf{S}_\varphi^d(T_\Sigma(X)) \to T_{\Sigma'}(\mathbf{S}_\varphi^d(X))$, is simply the inclusion map of the empty set $\mathbf{S}_\varphi^d(T_\Sigma(X))_{s'} = \emptyset$ into $T_{\Sigma'}(\mathbf{S}_\varphi^d(X))_{s'}$ for all $s' \in S' \setminus \varphi(S)$. Relying on the inductive definition of $T_\Sigma(X)$ in Definition 4 and the definition of $\mathbf{S}_\varphi^d$ by union, the family of maps $\overline{\varphi}_{X,s'} : \mathbf{S}_\varphi^d(T_\Sigma(X))_{s'} \to T_{\Sigma'}(\mathbf{S}_\varphi^d(X))_{s'}$ for all $s' \in \varphi(S)$ is inductively defined as follows:

Variables: $\overline{\varphi}_{X,\varphi(s)}(\ulcorner x \urcorner) := \ulcorner x \urcorner$ if $\ulcorner x \urcorner \in T_\Sigma(X)_s \subseteq \mathbf{S}_\varphi^d(T_\Sigma(X))_{\varphi(s)}$, $s \in S$;

Constants: $\overline{\varphi}_{X,\varphi(srt(\mathtt{c}))}(\ulcorner \mathtt{c}\langle\rangle \urcorner) := \ulcorner \varphi(\mathtt{c})\langle\rangle \urcorner$ if $\ulcorner \mathtt{c}\langle\rangle \urcorner \in T_\Sigma(X)_{srt(\mathtt{c})} \subseteq \mathbf{S}_\varphi^d(T_\Sigma(X))_{\varphi(srt(\mathtt{c}))}$, $\mathtt{c} \in F$, $in(\mathtt{c}) = \mathbf{0}_S$;

Operations: $\overline{\varphi}_{X,\varphi(srt(\mathtt{op}))}(\ulcorner \mathtt{op}\langle t_1, \ldots, t_n\rangle \urcorner) := \ulcorner \varphi(\mathtt{op})\langle \overline{\varphi}_{X,\varphi(s_1)}(t_1), \ldots, \overline{\varphi}_{X,\varphi(s_n)}(t_n)\rangle \urcorner$ if $\ulcorner \mathtt{op}\langle t_1, \ldots, t_n\rangle \urcorner \in T_\Sigma(X)_{srt(\mathtt{op})} \subseteq \mathbf{S}_\varphi^d(T_\Sigma(X))_{\varphi(srt(\mathtt{op}))}$, $\mathtt{op} \in F$, $n = |\bigcup in(\mathtt{op})| \geq 1$.

The conditions for signature morphisms ensure that the left-hand sides in the three defining equations above are indeed Σ'-terms over the S'-set $\mathbf{S}_\varphi^d(X)$ of variables. Note further, that $\ulcorner \langle \overline{\varphi}_{X,\varphi(s_1)}(t_1), \ldots, \overline{\varphi}_{X,\varphi(s_n)}(t_n)\rangle \urcorner = \mathbf{syn}(\mathbf{S}_\varphi^d(\boldsymbol{t}); \overline{\varphi}_X)$ for the assignment $\boldsymbol{t} = (\ulcorner t_1 \urcorner, \ldots, \ulcorner t_n \urcorner)$ in $T_\Sigma(X)^{in(\mathtt{op})}$ (see Remark 4).

Remark 11 (Induction). *Be aware, that we are not defining $\overline{\varphi}_X : \mathbf{S}_\varphi^d(T_\Sigma(X)) \to T_{\Sigma'}(\mathbf{S}_\varphi^d(X))$ by induction over $T_\Sigma(X)$ but by induction over $\mathbf{S}_\varphi^d(T_\Sigma(X))$! This is justified by the observation that the disjointedness of $T_\Sigma(X)$ ensures that the meta-level (!) inductive generation process of $T_\Sigma(X)$ as a set $\bigcup T_\Sigma(X)$ of strings of symbols is still the same for $\mathbf{S}_\varphi^d(T_\Sigma(X))$ since $\bigcup \mathbf{S}_\varphi^d(T_\Sigma(X)) = \bigcup T_\Sigma(X)$. Only the "typing" of the strings is changed (see Remark 3)!*

Due to the inductive definition of terms and the corresponding inductive definition of term translation, one can inductively prove the following lemma relying on the facts that $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$ is a functor and that composition of signature morphisms is defined componentwise.

Lemma 1 (Composition of Term Translations). *For any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ and the corresponding translations of terms $\overline{\varphi} : T_\Sigma \,; \mathbf{S}_\varphi^d \Rightarrow \mathbf{S}_\varphi^d \,; T_{\Sigma'} : Var_S \to \mathsf{Set}_d^{S'}$, $\overline{\psi} : T_{\Sigma'} \,; \mathbf{S}_\psi^d \Rightarrow \mathbf{S}_\psi^d \,; T_{\Sigma''} : Var_{S'} \to \mathsf{Set}_d^{S''}$ we do have:*

$$\overline{\varphi;\psi} = (\overline{\varphi}.\,\mathbf{S}_\psi^d);(\mathbf{S}_\varphi^d.\,\overline{\psi}) : T_\Sigma \,; \mathbf{S}_{\varphi;\psi}^d \Rightarrow \mathbf{S}_{\varphi;\psi}^d \,; T_{\Sigma''} : Var_S \to \mathsf{Set}_d^{S''}.$$

$$\begin{tikzcd}
Var_S \arrow[r, "T_\Sigma"] \arrow[d, "\mathbf{S}^d_\varphi"'] & \mathsf{Set}^S_d \arrow[d, "\mathbf{S}^d_\varphi"] \arrow[dl, Rightarrow, "\overline{\varphi}"'] \\
Var_{S'} \arrow[r, "T_{\Sigma'}"] \arrow[d, "\mathbf{S}^d_\psi"'] & \mathsf{Set}^{S'}_d \arrow[d, "\mathbf{S}^d_\psi"] \arrow[dl, Rightarrow, "\overline{\psi}"'] \\
Var_{S''} \arrow[r, "T_{\Sigma''}"] & \mathsf{Set}^{S''}_d
\end{tikzcd}
\qquad
\begin{tikzcd}
Var_S \arrow[r, "T_\Sigma"] \arrow[d, "\mathbf{S}^d_{\varphi;\psi}"'] & \mathsf{Set}^S_d \arrow[d, "\mathbf{S}^d_{\varphi;\psi}"] \arrow[dl, Rightarrow, "\overline{\varphi;\psi}"'] \\
Var_{S''} \arrow[r, "T_{\Sigma''}"] & \mathsf{Set}^{S''}_d
\end{tikzcd}$$

Remark 12 (Typing of Terms)**.** *This remark is triggered by a discussion at Bergen Language Design Laboratory (BLDL) in 2023. According to Definition 4, there is no "type information" encoded in terms. That is, the type (sort) of a term and its subterms has to be deduced from the types (sorts) of the variables and the input and output arities of the involved operation symbols declared in the corresponding signature Σ. Note, that expressions involving quantification carry, in contrast, type information due to Definition 5 and Remark 1.*

Let us consider signature morphisms only relaxing the type restrictions for predicate and operation symbols, i.e., signature morphisms $\varphi = (\varphi_{st}, id_P, id_F) : \Sigma \to \Sigma'$ between signatures $\Sigma = (S, P, F, ar, in, out)$ and $\Sigma' = (S', P, F, ar', in', out')$ with $\varphi_{st} : S \to S'$ a non-injective but surjective map. In this case, the natural transformation $\overline{\varphi} : T_\Sigma ; \mathbf{S}^d_\varphi \Rightarrow \mathbf{S}^d_\varphi ; T_{\Sigma'} : Var_S \to \mathsf{Set}^{S'}_d$ is not a real translation of terms but simply a family of inclusion S'-maps $\overline{\varphi}_X : \mathbf{S}^d_\varphi(T_\Sigma(X)) \hookrightarrow T_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ with X an S-set of variables.

In other words, for a given Σ'-term t there may be different signatures Σ and signature morphisms $\varphi = (\varphi_{st}, id_P, id_F) : \Sigma \to \Sigma'$ such that t is also a Σ-term!

4.2.3. Translation of Expressions

For each signature $\Sigma = (S, P, F, ar, in, out)$ we inductively defined in Definition 5 a map from Var_S into $|\mathsf{Set}|$. For convenience we can consider Var_S as a discrete subcategory of Set^S_d and use the functor notation $FE_\Sigma : Var_S \to \mathsf{Set}$ for this map.

Relying on the translation of terms, defined in Section 4.2.2, the translation of expressions for a signature morphism $\varphi : \Sigma \to \Sigma'$ is given by a natural transformation $\overline{\varphi} : FE_\Sigma \Rightarrow \mathbf{S}^d_\varphi ; FE_{\Sigma'} : Var_S \to \mathsf{Set}$. Be aware, that we use for both kinds of translation the same notation.

$$\begin{tikzcd}
Var_S \arrow[r, "FE_\Sigma"] \arrow[d, "\mathbf{S}^d_\varphi"'] & \mathsf{Set} \\
Var_{S'} \arrow[ur, "FE_{\Sigma'}"'] &
\arrow[from=1-1, to=2-2, phantom, "\Downarrow \overline{\varphi}"]
\end{tikzcd}$$

The family of maps $\overline{\varphi}_X : FE_\Sigma(X) \to FE_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ with X in Var_S is defined inductively and in parallel by the following assignments:

1. Atomic expressions:
   (a) Equation: $X \triangleright (t_1 = t_2) \mapsto \mathbf{S}^d_\varphi(X) \triangleright (\overline{\varphi}_{X,\varphi(s)}(t_1) = \overline{\varphi}_{X,\varphi(s)}(t_2))$ for any $s \in S$ and any $t_1, t_2 \in T_\Sigma(X)_s$.
   (b) Relational Atom: $X \triangleright \mathtt{p}\,\mathbf{syn}(\boldsymbol{t}) \mapsto \mathbf{S}^d_\varphi(X) \triangleright \varphi(\mathtt{p})\,\mathbf{syn}(\mathbf{S}^d_\varphi(\boldsymbol{t}); \overline{\varphi}_X)$ for any $\mathtt{p} \in P$ and any S-map $\boldsymbol{t} : ar(\mathtt{p}) \to T_\Sigma(X)$.
2. Everything: $X \triangleright \top\langle\rangle \mapsto \mathbf{S}^d_\varphi(X) \triangleright \top\langle\rangle$
3. Void: $X \triangleright \bot\langle\rangle \mapsto \mathbf{S}^d_\varphi(X) \triangleright \bot\langle\rangle$
4. Conjunction: $X \triangleright (Ex_1 \wedge Ex_2) \mapsto \mathbf{S}^d_\varphi(X) \triangleright (\overline{\varphi}_X(Ex_1) \wedge \overline{\varphi}_X(Ex_2))$
5. Disjunction: $X \triangleright (Ex_1 \vee Ex_2) \mapsto \mathbf{S}^d_\varphi(X) \triangleright (\overline{\varphi}_X(Ex_1) \vee \overline{\varphi}_X(Ex_2))$
6. Implication: $X \triangleright (Ex_1 \to Ex_2) \mapsto \mathbf{S}^d_\varphi(X) \triangleright (\overline{\varphi}_X(Ex_1) \to \overline{\varphi}_X(Ex_2))$
7. Negation: $X \triangleright \neg Ex \mapsto \mathbf{S}^d_\varphi(X) \triangleright \neg\, \overline{\varphi}_X(Ex)$

8. Quantification: $X \triangleright \exists(\mathbf{syn}(Y\backslash X) : Ex) \mapsto \mathbf{S}^d_\varphi(X) \triangleright \exists(\mathbf{syn}(\mathbf{S}^d_\varphi(Y)\backslash\mathbf{S}^d_\varphi(X)) : \overline{\varphi}_Y(Ex))$ and $X \triangleright \forall(\mathbf{syn}(Y\backslash X) : Ex) \mapsto \mathbf{S}^d_\varphi(X) \triangleright \forall(\mathbf{syn}(\mathbf{S}^d_\varphi(Y)\backslash\mathbf{S}^d_\varphi(X)) : \overline{\varphi}_Y(Ex))$ for any expression $Y \triangleright Ex$ and any proper S-inclusion $X \subset Y$.

For the case Quantification we have to bear in mind that $X \subset Y$ implies $\mathbf{S}^d_\varphi(X) \subset \mathbf{S}^d_\varphi(Y)$ and $\mathbf{S}^d_\varphi(Y \setminus X) = \mathbf{S}^d_\varphi(Y) \setminus \mathbf{S}^d_\varphi(X)$. By means of Lemma 1 one can inductively prove the following lemma relying on the facts that $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$ is a functor and that composition of signature morphisms is defined componentwise.

Lemma 2 (Composition of Expression Translations)**.** *For any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ and the corresponding translations of expressions $\overline{\varphi} : FE_\Sigma \Rightarrow \mathbf{S}^d_\varphi\,;FE_{\Sigma'} : Var_S \to \mathsf{Set}$ and $\overline{\psi} : FE_{\Sigma'} \Rightarrow \mathbf{S}^d_\psi\,;FE_{\Sigma''} : Var_{S'} \to \mathsf{Set}$ we do have:*

$$\overline{\varphi;\psi} = \overline{\varphi};(\mathbf{S}^d_\varphi.\overline{\psi}) : FE_\Sigma \Rightarrow \mathbf{S}^d_{\varphi;\psi}\,;FE_{\Sigma''} : Var_S \to \mathsf{Set}.$$

$$\begin{tikzcd}
Var_S \arrow[d, "\mathbf{S}^d_\varphi"'] \arrow[dr, "FE_\Sigma"] & \\
Var_{S'} \arrow[r, "FE_{\Sigma'}"] \arrow[d, "\mathbf{S}^d_\psi"'] \arrow[u, phantom, shift left=4ex, "\overline{\varphi}\Downarrow"] & \mathsf{Set} \\
Var_{S''} \arrow[ur, "FE_{\Sigma''}"'] \arrow[u, phantom, shift left=4ex, "\overline{\psi}\Downarrow"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
Var_S \arrow[dd, "\mathbf{S}^d_{\varphi;\psi}"'] \arrow[dr, "FE_\Sigma"] & \\
\arrow[r, phantom, "\overline{\varphi;\psi}\Downarrow"] & \mathsf{Set} \\
Var_{S''} \arrow[ur, "FE_{\Sigma''}"'] &
\end{tikzcd}$$

4.2.4. Translation of Syntax vs. Reduct of Structures

For any signature morphism $\varphi : \Sigma \to \Sigma'$ we do have now, on the semantic side, a corresponding reduct functor $\mathbf{Str}(\varphi) = (\_\upharpoonright_\varphi) : \mathsf{Str}(\Sigma') \to \mathsf{Str}(\Sigma)$ and, on the syntactic side, a corresponding translation $\overline{\varphi} : T_\Sigma\,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{S}^d_\varphi\,;T_{\Sigma'} : Var_S \to \mathsf{Set}^{S'}_d$ of terms as well as a corresponding translation $\overline{\varphi} : FE_\Sigma \Rightarrow \mathbf{S}^d_\varphi\,;FE_{\Sigma'} : Var_S \to \mathsf{Set}$ of expressions in the opposite direction.

The inductive definitions of terms and expressions and their syntactic translations ensure that the compatibility between the semantics of the basic syntactic entities operation symbol, predicate symbol and their syntactic translations, as described in Definition 8, lifts up to the derived syntactic entities term and expression.

For a given Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$ we consider its φ-reduct, i.e., the Σ-structure $\mathcal{A}'\!\upharpoonright_\varphi = (\mathbf{R}_\varphi(A'), P^{\mathcal{A}'\upharpoonright_\varphi}, F^{\mathcal{A}'\upharpoonright_\varphi})$ as defined in Definition 8. First, we discuss the compatibility between the semantics of terms and their syntactic translation. According to (4), any Σ-substitution declaration $\boldsymbol{t} : Y \to T_\Sigma(X)$ defines a derived operation $\boldsymbol{t}^{\mathcal{A}'\upharpoonright_\varphi} : \mathbf{R}_\varphi(A')^X \to \mathbf{R}_\varphi(A')^Y$ in $\mathcal{A}'\!\upharpoonright_\varphi$. By means of $\overline{\varphi} : T_\Sigma\,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{S}^d_\varphi\,;T_{\Sigma'}$ we can translate the Σ-substitution declaration $\boldsymbol{t} : Y \to T_\Sigma(X)$ into a Σ'-substitution declaration $\mathbf{S}^d_\varphi(\boldsymbol{t});\overline{\varphi}_X : \mathbf{S}^d_\varphi(Y) \to T_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ and obtain a corresponding derived operation $(\mathbf{S}^d_\varphi(\boldsymbol{t});\overline{\varphi}_X)^{\mathcal{A}'} : A'^{\mathbf{S}^d_\varphi(X)} \to A'^{\mathbf{S}^d_\varphi(Y)}$ in $\mathcal{A}'$. The bijections in (7) establish a correspondence between the two derived operations assuring that $\boldsymbol{t}^{\mathcal{A}'\upharpoonright_\varphi}$ can be considered as the φ-reduct of $(\mathbf{S}^d_\varphi(\boldsymbol{t});\overline{\varphi}_X)^{\mathcal{A}'}$.

Proposition 1 (Derived Operation Reduct)**.** *For any signature morphism $\varphi : \Sigma \to \Sigma'$, any Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$ and any Σ-substitution declaration $\boldsymbol{t} : Y \to T_\Sigma(X)$ we have*

$$\mathbf{B}_\varphi;\boldsymbol{t}^{\mathcal{A}'\upharpoonright_\varphi} = (\mathbf{S}^d_\varphi(\boldsymbol{t});\overline{\varphi}_X)^{\mathcal{A}'};\mathbf{B}_\varphi\,.$$

$$\begin{tikzcd}
\mathcal{A}'^{\mathbf{S}^d_\varphi(X)} \arrow[r, "(\mathbf{S}^d_\varphi(t);\overline{\varphi}_X)^{\mathcal{A}'}"] \arrow[d, "\mathbf{B}_\varphi"'] \arrow[dr, phantom, "="] & \mathcal{A}'^{\mathbf{S}^d_\varphi(Y)} \arrow[d, "\mathbf{B}_\varphi"] \\
\mathbf{R}_\varphi(A')^X \arrow[r, "t^{\mathcal{A}'\upharpoonright_\varphi}"] & \mathbf{R}_\varphi(A')^Y
\end{tikzcd}$$

Second, we discuss the compatibility between the semantics of expressions and their syntactic translation. For any Σ-expression Ex on an S-set X of variables its semantics in the reduct $\mathcal{A}'\upharpoonright_\varphi$ is given by the derived predicate $[\![\, Ex \,]\!]^{\mathcal{A}'\upharpoonright_\varphi}_X \subseteq \mathbf{R}_\varphi(A')^X$. By means of $\overline{\varphi} : FE_\Sigma \Rightarrow \mathbf{S}^d_\varphi\, ; FE_{\Sigma'}$ we can translate Ex into a Σ'-expression $\overline{\varphi}_X(Ex)$ on the S'-set $\mathbf{S}^d_\varphi(X)$ of variables and obtain a corresponding derived predicate $[\![\, \overline{\varphi}_X(Ex) \,]\!]^{\mathcal{A}'}_{\mathbf{S}^d_\varphi(X)} \subseteq \mathcal{A}'^{\mathbf{S}^d_\varphi(X)}$ in $\mathcal{A}'$. The bijections in (7) establish a correspondence between the two derived predicates assuring that $[\![\, Ex \,]\!]^{\mathcal{A}'\upharpoonright_\varphi}_X$ can be considered as the φ-reduct of $[\![\, \overline{\varphi}_X(Ex) \,]\!]^{\mathcal{A}'}_{\mathbf{S}^d_\varphi(X)}$.

Proposition 2 (Derived Predicate Reduct)**.** *For any signature morphism $\varphi : \Sigma \to \Sigma'$, any Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$ and any Σ-expression Ex on an S-set X of variables the bijection $\mathbf{B}_\varphi : \mathcal{A}'^{\mathbf{S}^d_\varphi(X)} \to \mathbf{R}_\varphi(A')^X$ restricts to a bijection $\mathbf{B}_\varphi : [\![\, \overline{\varphi}_X(Ex) \,]\!]^{\mathcal{A}'}_{\mathbf{S}^d_\varphi(X)} \to [\![\, Ex \,]\!]^{\mathcal{A}'\upharpoonright_\varphi}_X$.*

$$\begin{tikzcd}
{[\![\, \overline{\varphi}_X(Ex) \,]\!]^{\mathcal{A}'}_{\mathbf{S}^d_\varphi(X)}} \arrow[r, "\subseteq"] \arrow[d, "\mathbf{B}_\varphi"'] \arrow[dr, phantom, "="] & \mathcal{A}'^{\mathbf{S}^d_\varphi(X)} \arrow[d, "\mathbf{B}_\varphi"] \\
{[\![\, Ex \,]\!]^{\mathcal{A}'\upharpoonright_\varphi}_X} \arrow[r, "\subseteq"] & \mathbf{R}_\varphi(A')^X
\end{tikzcd}$$

If we use the more traditional notation $\boldsymbol{a} \models^\Sigma_{\mathcal{A}} X \triangleright Ex$ to indicate that $\boldsymbol{a} \in [\![\, Ex \,]\!]^{\mathcal{A}}_X$ for an assignment $\boldsymbol{a} : X \to A$ in a Σ-structure $\mathcal{A}$, we get the following equivalent formulation of the statement in Proposition 2: For any signature morphism $\varphi : \Sigma \to \Sigma'$, any Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$, any Σ-expression Ex on an S-set X of variables and any assignment $\boldsymbol{a}' : \mathbf{S}^d_\varphi(X) \to A'$ and thus $\mathbf{B}_\varphi(\boldsymbol{a}') : X \to \mathbf{R}_\varphi(A')$ it holds that

$$\mathbf{B}_\varphi(\boldsymbol{a}') \models^\Sigma_{\mathcal{A}'\upharpoonright_\varphi} X \triangleright Ex \quad \text{iff} \quad \boldsymbol{a}' \models^{\Sigma'}_{\mathcal{A}'} \mathbf{S}^d_\varphi(X) \triangleright \overline{\varphi}_X(Ex). \tag{12}$$

Following the tradition in abstract model theory, we may call this the **satisfaction condition for expressions and assignments**.

4.2.5. Traditional Institution of First-Order Logic

We obtain the traditional institution $\mathcal{FOL}$ of first-order logic, as presented in [16,17] for example, if we restrain our results to empty sets of variables only.

We start with the category Sig of all many-sorted signatures and signature morphisms. The "model functor", in the sense of institutions, is the functor $\mathbf{Str} : \mathsf{Sig}^{op} \to \mathsf{CAT}$ given by the assignments $\Sigma \mapsto \mathsf{Str}(\Sigma)$ and $\varphi \mapsto \mathbf{Str}(\varphi) = (\_\upharpoonright_\varphi)$.

For each signature $\Sigma = (S, P, F, ar, in, out)$ we consider only the empty S-set $\mathbf{0}_S$ of variables. The Σ-sentences are exactly all closed Σ-expressions $\mathbf{0}_S \triangleright Ex$ (see Remark 7). For any signature morphism $\varphi : \Sigma \to \Sigma'$ the corresponding translation of sentences is given by the map $\overline{\varphi}_{\mathbf{0}_S} : FE_\Sigma(\mathbf{0}_S) \to FE_{\Sigma'}(\mathbf{0}_{S'})$. Keep in mind that $\mathbf{S}^d_\varphi(\mathbf{0}_S) = \mathbf{0}_{S'}$. Lemma 2 ensures that the assignments $\Sigma \mapsto FE_\Sigma(\mathbf{0}_S)$ and $(\varphi : \Sigma \to \Sigma') \mapsto (\overline{\varphi}_{\mathbf{0}_S} : FE_\Sigma(\mathbf{0}_S) \to FE_{\Sigma'}(\mathbf{0}_{S'}))$ define a sentence functor $\mathbf{cFE} : \mathsf{Sig} \to \mathsf{Set}$ in the sense of institutions.

We say that a closed Σ-expression $\mathbf{0}_S \triangleright Ex$ is satisfied in a Σ-structure $\mathcal{A}$, $\mathcal{A} \models^\Sigma \mathbf{0}_S \triangleright Ex$ in symbols if, and only if, $!_A \models^\Sigma_{\mathcal{A}} \mathbf{0}_S \triangleright Ex$, i.e., $[\![\, Ex \,]\!]^{\mathcal{A}}_{\mathbf{0}_S} = \{!_A\}$ (compare Remark 9).

For any signature morphism $\varphi : \Sigma \to \Sigma'$ and any Σ'-structure $\mathcal{A}' = (A', P^{\mathcal{A}'}, F^{\mathcal{A}'})$ we do have $\mathbf{S}^d_\varphi(\mathbf{0}_S) = \mathbf{0}_{S'}$ and $\mathbf{B}_\varphi(!_{A'}) = !_{A'\restriction_\varphi} = !_{\mathbf{R}_\varphi(A')} : \mathbf{0}_S \to \mathbf{R}_\varphi(A')$ thus (12) specializes for $X = \mathbf{0}_S$ to the statement

$$!_{A'\restriction_\varphi} \models^{\Sigma}_{\mathcal{A}'\restriction_\varphi} \mathbf{0}_S \rhd Ex \quad \text{iff} \quad !_{A'} \models^{\Sigma'}_{\mathcal{A}'} \mathbf{0}_{S'} \rhd \overline{\varphi}_{\mathbf{0}_S}(Ex). \tag{13}$$

This means, that the satisfaction condition holds indeed in $\mathcal{FOL} = (\mathsf{Sig}, \mathbf{cFE}, \mathbf{Str}, \models)$.

A crucial fact is that closed expressions, their translations and their semantics are constructed via a long detour through the realm of "open formulas" thus the proof of the satisfaction condition in (13), for example, requires to prove (at least implicitly) general statements like in (12). One motivation to develop Logics of Statements in Context was to establish a "conceptual and technological space" where all the general results concerning "open formulas" and variable assignments can be formalized, developed and presented in their full extent and beauty. The anticipated "conceptual and technological space" is also meant to become the home for deduction since deduction of first-order formulas takes essentially place in the realm of "open formulas".

It is well-known that empty sorts potentially cause the problem that deduction rules for one-sorted first-order logic are, in general, not sound for many-sorted deduction [17,19]. As a matter of fact, the simple concept of expression turns out to be not sophisticated enough to describe, formalize and analyze first-order deduction and the problems concerning many-sorted first-order deduction. In Logics of Statements in Context we introduce therefore a new conceptual layer of contexts, between the layer of sets of variables and the layer of carrier sets, and we are lifting up the concept of expression to the more flexible concept of "statement in context".

5. First-Order Logics of Statements in Context

5.1. Institutions of Statements

In this subsection we construct for each signature $\Sigma = (S, P, F, ar, in, out)$ an institution of statements $\mathcal{FL}_\Sigma = (\mathsf{Cxt}_S, \mathbf{Stm}_\Sigma, \mathbf{Int}_\Sigma, \models^\Sigma)$ combining the constructions of Institutions of Statements and of Institutions of Equations in [1]. Compared to the general case in [1], we consider only the variant with the full spectrum FE_Σ of first-order expressions and with the whole category $\mathsf{Str}(\Sigma)$ of Σ-structures as semantics $\mathsf{Sem}(\Sigma)$. Therefore, we only need a simplified version of the construction of institution of statements as visualized in Figure 1.

$$\begin{tikzcd}
 & & \mathsf{Var}_S \arrow[r, dashed, "constr.+", "choice"'] & \Sigma \arrow[r, dashed, "constr."] \arrow[dd] & FE_\Sigma \arrow[r] & Stm_\Sigma(K) \arrow[dd, leftrightarrow, "constr."', "\models^\Sigma_K"] \\
\bullet \arrow[r, dotted, "choice"] & \mathsf{Base}_S \arrow[ur, dotted, "choice"] \arrow[r, dotted, "choice"] \arrow[dr, dotted, "choice"'] & \mathsf{Cxt}_S \arrow[rr, squiggly, "varies"] & & K \arrow[ur] \arrow[dr] \arrow[u, phantom, "constr."] \arrow[dl, phantom, "constr."] & \\
 & & \mathsf{Carr}_S \arrow[r] \arrow[ur, phantom, "constr."] & \mathsf{Str}(\Sigma) \arrow[rr] & & \mathsf{Int}_\Sigma(K)
\end{tikzcd}$$

Figure 1. Stepwise construction of an Institution of First-Order Statements.

The "base category" and the "category of (potential) carriers" are simply the category of S-sets: $\mathsf{Base}_S = \mathsf{Carr}_S := \mathsf{Set}^S$. At this stage, we do not consider the translation of expressions along substitutions $\boldsymbol{t} : X \to T_\Sigma(Y)$ and thus not along inclusions $X \subset Y$ either. Those translations are discussed in Section 6. Therefore, we choose as category Var_S of "variable declarations" simply the set Var_S considered as a discrete category. $\mathsf{Str}(\Sigma)$ is the category of all Σ-structures, as defined in Section 3.1, and $FE_\Sigma : Var_S \to \mathsf{Set}$ is the Var_S-indexed family of sets of first-order Σ-expressions, as defined in Definition 5.

It remains to define the "category of contexts" Cxt_S. In traditional First-Order Logic contexts are not considered at all thus we could choose a cautious step and define Cxt_S as the extension of Var_S by all inclusions between S-sets of variables. Those inclusions are, at least, implicitly present in traditional

First-Order Logic! We drop this step for three reasons. First, we would just blow up the paper by "copying and paste" without providing new theoretical substance. Second, it would be too annoying always to underline and discuss why the same S-set X of variables plays, methodological seen, two different roles depending on if we consider X as an object in Var_S or as an object in Cxt_S, respectively. The methodological distinction between these two roles becomes more naturally evident if we consider arbitrary contexts. Third, it is convenient to consider arbitrary contexts from the very beginning to be prepared for the discussion of "diagrams" [2,3,17] in Section 7.

In Section 4 we learned that the translation of syntactic entities along signature morphisms becomes only functorial if the definition of syntactic entities is based on disjoint S- sets. Therefore, we define the category of contexts to be the category of disjoint S-sets: $\mathsf{Cxt}_S := \mathsf{Set}^S_d$. Contexts are the **abstract signatures** in an institution of statements. Keep in mind that $\mathsf{Var}_S \sqsubset \mathsf{Cxt}_S \sqsubset \mathsf{Base}_S$ by definition.

Since the interplay of predicates and equations is encapsulated in the construction of FE_Σ , the general construction of Institutions of Statements in [1] can easily be adapted for first-order logic with predicates and operations. To make the paper sufficiently self-contained and accessible, we give a concise presentation of the adapted definitions, constructions and results.

5.1.1. Institutions of Statements: Sentence and Model Functor

Statements in context are the **sentences** in an institutions of statements.

Definition 10 (Statement). *A* ***Σ-statement*** (X, Ex, γ) ***in a context*** $K \in |\mathsf{Cxt}_S|$ *is given by a Σ-expression Ex in $FE_\Sigma(X)$ and a* ***binding S-map*** $\gamma\colon X \to K$. (X, Ex, γ) *is called* ***atomic*** *if Ex is an atomic expression, i.e., an Equation or a Relational Atom.*

By $Stm_\Sigma(K)$, we denote the set of all Σ-statements in K and by $\Upsilon_K : Stm_\Sigma(K) \to Var_S$ the obvious projection map.

Remark 13 (General statements and closed formulas). *For any closed Σ-expression $\mathbf{0}_S \triangleright Ex$ (see Remarks 7 and 9) there is a unique initial morphism $\gamma =!_K : \mathbf{0}_S \to K$; thus, we have $(\mathbf{0}_S, Ex, !_K) \in Stm_\Sigma(K)$ for any context K and all closed Σ-expressions $\mathbf{0}_S \triangleright Ex$ in $FE_\Sigma(\mathbf{0}_S)$. We call $(\mathbf{0}_S, Ex, !_K)$ a* ***general statement in*** *K.*

From all the general statements $(\mathbf{0}_S, Ex, !_K)$ sharing the same closed expression $\mathbf{0}_S \triangleright Ex$, only the general statement $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ in the empty context $\mathbf{0}_S$ is the proper formal counterpart of traditional ***closed formulas*** *in institutions of statements in context!*

Example 7 (Context and Statement). *For the sample signature Σ in Example 1, we define a context K: $K_{prs} := \{Anna, Michael, Stina, Dora, Heinz, Gabi, Uwe\}$ is chosen to be a finite set of* symbolic literals *in the sense of PROLOG. We distinguish clearly between sets of variables, carriers of structures and contexts, but allow that the same entity can play different roles (see also Section 7.2). In this spirit, we choose K_{data} to be the set of all dates from 01.01.2000 to 31.12.2020 and $K_{nat} := \{n \mid n \in \mathbb{N}, 1 \leq n \leq 600\}$.*

In examples we adapt the encoding of assignments from Remark 2 and use for statements (X, Ex, γ) the shorthand notation $Ex(\gamma_1, \ldots, \gamma_n)$ accepting the possible loss that we can not reconstruct from $Ex(\gamma_1, \ldots, \gamma_n)$ the whole S-set X of variables. Instead of expressions Ex, we will also use auxiliary names for expressions as introduced in Example 4, for example. If Ex is a "reborn predicate symbol" (see Remark 5) we will use the traditional notation $\mathtt{p}(\gamma_1, \ldots, \gamma_n)$ instead of $\mathtt{p}\langle v_1^{ar(\mathtt{p})}, \ldots, v_n^{ar(\mathtt{p})}\rangle(\gamma_1, \ldots, \gamma_n)$.

$(\mathtt{born}\langle x_1\rangle = x_2)(Uwe, 19.10.1958)$ *is not a legal Σ-statement in context K simply because* 19.10.1958 *is not contained in K_{date}.* $(\mathtt{born}\langle x_1\rangle = \mathtt{sp}\langle\rangle)(Uwe)$*, however, is an atomic legal Σ-statement in K. Attention, we are not using a notation like $(\mathtt{born}\langle Uwe\rangle = \mathtt{sp}\langle\rangle)$, instead of $(\mathtt{born}\langle x_1\rangle = \mathtt{sp}\langle\rangle)(Uwe)$, since Uwe is neither a variable nor a constant symbol, i.e., $(\mathtt{born}\langle Uwe\rangle = \mathtt{sp}\langle\rangle)$ is not a Σ-expression. To say it differently: In $(\mathtt{born}\langle Uwe\rangle = \mathtt{sp}\langle\rangle)$ syntax (predicate symbols, expressions, auxiliary names) and semantics (assignments) are not separated.*

$\mathtt{par}(Uwe, Heinz, Dora)$, $\mathtt{par}(Anna, Uwe, Gabi)$, $\mathtt{par}(Michael, Uwe, Gabi)$ *are atomic Σ-statements in K and from the last two Σ-statements we could deduce $\mathtt{sbl}(Anna)$ and $\mathtt{sbl}(Michael)$.*

$\mathtt{sbl}(Gabi), \mathtt{hst}(Stina), \mathtt{sbl}(Uwe), (\mathtt{born}\langle x_1\rangle = \mathtt{pl}\langle \mathtt{born}\langle x_2\rangle, x_3\rangle)(Michael, Anna, 600)$ *and* $(\mathtt{born}\langle x_1\rangle = x_2)(Anna, 14.03.2009)$ *are other* Σ*-statements in context* K*.*

Any morphism $\mu : K \to G$ in Cxt_S induces a map $\mathbf{Stm}_\Sigma(\mu) : Stm_\Sigma(K) \to Stm_\Sigma(G)$ defined by simple post-composition for all statements (X, Ex, γ) in K:

$$\begin{tikzcd} & X \rhd Ex \arrow[dl, "\gamma"', ""{name=L}] \arrow[dr, "\gamma;\mu", ""'{name=R}] & \\ K \arrow[rr, "\mu"'] & & G \arrow[from=L, to=R, mapsto, bend right] \end{tikzcd} \qquad \mathbf{Stm}_\Sigma(\mu)(X, Ex, \gamma) := (X, Ex, \gamma; \mu). \tag{14}$$

It is easy to show that the assignments $K \mapsto Stm_\Sigma(K)$ and $\mu \mapsto \mathbf{Stm}_\Sigma(\mu)$ provide a functor $\mathbf{Stm}_\Sigma : \mathsf{Cxt}_S \to \mathsf{Set}$. This is the **sentence functor** of the institution $\mathcal{FL}_\Sigma$.

Interpretations of contexts are the **models** in an institution of statements.

Definition 11 (Context interpretations)**.** *A* ***Σ-interpretation*** $(\iota, \mathcal{A})$ ***of a context*** $K \in |\mathsf{Cxt}_S|$ *is given by a* Σ*-structure* $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ *in* $\mathsf{Str}(\Sigma)$ *and an* S*-map* $\iota : K \to A$*.*

A morphism $\varsigma : (\iota, \mathcal{A}) \to (\vartheta, \mathcal{B})$ *between* Σ*-interpretations of* K *is given by a* Σ*-homomorphism* $\varsigma : \mathcal{A} \to \mathcal{B}$ *such that* $\iota; \varsigma = \vartheta$ *for the underlying* S*-map* $\varsigma : A \to B$*.*

$$\begin{tikzcd} & K \arrow[dl, "\iota"'] \arrow[dr, "\vartheta"] \arrow[d, phantom, "="] & \\ A \arrow[rr, "\varsigma"] & {} & B \end{tikzcd}$$

For any context K *in* Cxt_S*, we denote by* $\mathsf{Int}_\Sigma(K)$ *the category of all* Σ*-interpretations of* K *and all morphisms between them and by* $\Pi_K : \mathsf{Int}_\Sigma(K) \to \mathsf{Str}(\Sigma)$ *the obvious projection functor.*

Remark 14 (Split Opfibration)**.** *It is easy to see that the functor* $\Pi_K : \mathsf{Int}_\Sigma(K) \to \mathsf{Str}(\Sigma)$ *is a split opfibration with the opcartesian arrow* $\varsigma : (\iota, \mathcal{A}) \to (\iota; \varsigma, \mathcal{B})$ *for a* Σ*-homomorphism* $\varsigma : \mathcal{A} \to \mathcal{B}$ *and an interpretation* $(\iota, \mathcal{A})$ *of context* K *in the* Σ*-structure* $\mathcal{A}$ *[8]. Be aware that all morphisms in* $\mathsf{Int}_\Sigma(K)$ *are opcartesion arrows, that is, for any* Σ*-structure* $\mathcal{A}$*, the corresponding* ***fiber over*** $\mathcal{A}$*, i.e., the subcategory of* $\mathbf{Int}_\Sigma(K)$ *given by all interpretations of* K *in* $\mathcal{A}$ *and all context morphisms* $id_{\mathcal{A}} : (\iota, \mathcal{A}) \to (\iota, \mathcal{A})$*, is a discrete category representing the hom-set* $\mathsf{Set}^S(K, A)$*.*

Note finally, that $\Pi_{\mathbf{0}_S} : \mathsf{Int}_\Sigma(\mathbf{0}_S) \to \mathsf{Str}(\Sigma)$ *is an isomorphism.*

Any morphism $\mu : K \to G$ in Cxt_S induces a functor $\mathbf{Int}_\Sigma(\mu) : \mathsf{Int}_\Sigma(G) \to \mathsf{Int}_\Sigma(K)$ defined by simple pre-composition for all Σ-interpretations $(\vartheta, \mathcal{B})$ of G:

$$\mathbf{Int}_\Sigma(\mu)(\vartheta, \mathcal{B}) := (\mu; \vartheta, \mathcal{B}). \tag{15}$$

$$\begin{tikzcd} K \arrow[rr, "\mu"] \arrow[dr, "\mu;\vartheta"', ""{name=L}] & & G \arrow[dl, "\vartheta", ""'{name=R}] \\ & B \arrow[from=R, to=L, mapsto, bend right] & \end{tikzcd} \qquad \begin{tikzcd} K \arrow[r, "\mu"] \arrow[d, "\mu;\iota"'] \arrow[dr, "\mu;\vartheta"' near start] & G \arrow[d, "\vartheta"] \arrow[dl, "\iota" near start] \\ A \arrow[r, "\varsigma"'] & B \end{tikzcd} \qquad \begin{tikzcd} \mathsf{Int}_\Sigma(K) \arrow[dr, "\Pi_K"'] & & \mathsf{Int}_\Sigma(G) \arrow[ll, "\mathbf{Int}_\Sigma(\mu)"'] \arrow[dl, "\Pi_G"] \\ & \mathsf{Str}(\Sigma) \arrow[u, phantom, "="] & \end{tikzcd}$$

For any morphism $\varsigma : (\iota, \mathcal{A}) \to (\vartheta, \mathcal{B})$ between two Σ-interpretations of G the same underlying S-map $\varsigma : A \to B$ establishes a morphism $\mathbf{Int}_\Sigma(\mu)(\varsigma) := \varsigma : (\mu; \iota, \mathcal{A}) \to (\mu; \vartheta, \mathcal{B})$ between the corresponding two Σ-interpretations of K, thus we have

$$\mathbf{Int}_\Sigma(\mu); \Pi_K = \Pi_G : \mathsf{Int}_\Sigma(G) \to \mathsf{Str}(\Sigma) \quad \text{(see the diagram above on the right).} \tag{16}$$

One can easily validate that the assignments $K \mapsto \mathsf{Int}_\Sigma(K)$ and $\mu \mapsto \mathbf{Int}_\Sigma(\mu)$ define a functor $\mathbf{Int}_\Sigma : \mathsf{Cxt}_S^{op} \to \mathsf{CAT}$. This is the **model functor** of the institution $\mathcal{FL}_\Sigma$.

5.1.2. Institutions of Statements: Satisfaction Relation and Satisfaction Condition

The last two steps, in establishing an institution, are the definition of a *satisfaction relations* and the proof of the so-called *satisfaction condition*. The satisfaction relations are simply provided by the semantics of expressions, as described in Definition 6.

Definition 12 (Satisfaction Relation). *For any context $K \in |\mathsf{Cxt_S}|$, any Σ-statement (X, Ex, γ) in K and any Σ-interpretation $(\iota, \mathcal{A})$ of context K we define:*

$$(\iota, \mathcal{A}) \models^{\Sigma}_{K} (X, Ex, \gamma) \quad \mathit{iff} \quad \gamma;\iota \models^{\Sigma}_{\mathcal{A}} X \rhd Ex \quad (\,i.e. \quad \gamma;\iota \in [\![\, Ex \,]\!]^{\mathcal{A}}_{X}\,) \tag{17}$$

$$\begin{tikzcd} & K \arrow[dl, "\iota"'] & \\ A & & X \rhd Ex \arrow[ul, "\gamma"'] \arrow[ll, "\gamma;\iota"'] \end{tikzcd}$$

Remark 15 (Validity of Closed Formulas). *If $X = K = \mathbf{0}_S$, we do have for any Σ-structure $\mathcal{A} = (A, P^{\mathcal{A}}, F^{\mathcal{A}})$ in $\mathsf{Str}(\Sigma)$ exactly one interpretation $(!_A, \mathcal{A})$ thus for any closed formula $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ (see Remark 13) $(!_A, \mathcal{A}) \models^{\Sigma}_{\mathbf{0}_S} (\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ means nothing but that the closed formula $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ is* valid *in $\mathcal{A}$ in the traditional sense.*

Moreover, the validity of closed formulas is "semantically context independent" in the following sense: For any context K, any Σ-structure $\mathcal{A}$, any Σ-interpretation $(\iota, \mathcal{A})$ of K in $\mathcal{A}$ and any closed expressions $\mathbf{0}_S \rhd Ex$, we have due to the uniqueness if intitial morphisms:

$$(\iota, \mathcal{A}) \models^{\Sigma}_{K} (\mathbf{0}_S, Ex, !_K) \quad \mathit{iff} \quad !_K;\iota = !_A = id_{\mathbf{0}_S};!_A \in [\![\, Ex \,]\!]^{\mathcal{A}}_{\mathbf{0}_S} \quad \mathit{iff} \quad (!_A, \mathcal{A}) \models^{\Sigma}_{\mathbf{0}_S} (\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$$

Example 8 (Interpretation and Satisfaction). *For the sample context K in Example 7 and the sample Σ-structure $\mathcal{A}$ in Example 2, we consider the "author intended interpretation" $(\iota, \mathcal{A})$ of K in $\mathcal{A}$ with the S-map $\iota : K \to \mathcal{A}$ defined as follows: ι_{date} is the inclusion map from K_{date} into A_{date}, ι_{nat} the inclusion map from K_{nat} into $A_{nat} = \mathbb{N}^+$ while $\iota_{prs} : K_{prs} \to A_{prs}$ assigns to each symbolic literal in K_{prs} the corresponding "real person" in A_{prs} from the family of the author. We discuss the satisfaction of the sample Σ-statements from Example 7 by $(\iota, \mathcal{A})$.*

$(\mathtt{born}\langle x_1 \rangle = \mathtt{sp}\langle\rangle)(Uwe)$ *is satisfied by $(\iota, \mathcal{A})$ since* $\mathtt{born}^{\mathcal{A}}(\iota_{prs}(Uwe)) = 19.10.1958$ *and* $\mathtt{sp}^{\mathcal{A}}() = 19.10.1958$*. The atomic Σ-statements $\mathtt{par}(Uwe, Heinz, Dora)$, $\mathtt{par}(Anna, Uwe, Gabi)$ and $\mathtt{par}(Michael, Uwe, Gabi)$ are satisfied by $(\iota, \mathcal{A})$ since they reflect, due to the definition of $\mathtt{par}^{\mathcal{A}}$, the "real life" family situation of the author. In such a way, the definition of Σ-expressions and their semantics, ensures that also the two Σ-statements $\mathtt{sbl}(Anna)$ and $\mathtt{sbl}(Michael)$ are satisfied by $(\iota, \mathcal{A})$, i.e., $\iota_{prs}(Anna), \iota_{prs}(Michael) \in [\![\, \mathtt{sbl} \,]\!]^{\mathcal{A}}_{(x_1 : prs)}$.*

$\mathtt{sbl}(Gabi)$ is not satisfied by $(\iota, \mathcal{A})$ since the "real person" $\iota_{prs}(Gabi)$ is an only child. $\mathtt{hst}(Stina)$ is true in "real life", however, not satisfied by $(\iota, \mathcal{A})$ since the mother of the "real person" $\iota_{prs}(Stina)$ is born in Venezuela. $\mathtt{sbl}(Uwe)$ is satisfied by $(\iota, \mathcal{A})$ even if the only sibling of the author is not represented by a symbolic literal in K_{prs}. $(\mathtt{born}\langle x_1 \rangle = x_2)(Anna, 14.03.2009)$ and $(\mathtt{born}\langle x_1 \rangle = \mathtt{pl}\langle \mathtt{born}\langle x_2 \rangle, x_3 \rangle)(Michael, Anna, 600)$ are, finally, satisfied by $(\iota, \mathcal{A})$ since $\mathtt{born}^{\mathcal{A}}(\iota_{prs}(Anna)) = 14.03.2009$ *and* $\mathtt{born}^{\mathcal{A}}(\iota_{prs}(Michael)) = 04.11.2010$.

After we developed everything in a systematic modular way, we obtain the satisfaction condition nearly “for free”.

Corollary 1 (Satisfaction condition). *For any morphism $\mu : K \to G$ in $\mathsf{Cxt_S}$, any Σ-statement (X, Ex, γ) in context K and any interpretation $(\vartheta, \mathcal{A})$ of context G in a Σ-structure $\mathcal{B}$, we have:*

$$\mathbf{Int}_{\Sigma}(\mu)(\vartheta, \mathcal{A}) \models^{\Sigma}_{K} (X, Ex, \gamma) \quad \mathit{iff} \quad (\vartheta, \mathcal{A}) \models^{\Sigma}_{G} \mathbf{Stm}_{\Sigma}(\mu)(X, Ex, \gamma). \tag{18}$$

$$\begin{tikzcd}
K \arrow[d, "\mu"'] & (\mu;\vartheta,\mathcal{A}) & \models^{\Sigma}_{K} & (X,Ex,\gamma) \arrow[d, mapsto, "\mathbf{Stm}_{\Sigma}(\mu)"] \\
G & (\vartheta,\mathcal{A}) \arrow[u, mapsto, "\mathbf{Int}_{\Sigma}(\mu)"] & \models^{\Sigma}_{G} & (X,Ex,\gamma;\mu)
\end{tikzcd}
\qquad
\begin{tikzcd}
 & K \arrow[dl, "\mu;\vartheta"'] \arrow[dd, "\mu"] & \\
A & & X \rhd Ex \arrow[ul, "\gamma"'] \arrow[dl, "\gamma;\mu"] \\
 & G \arrow[ul, "\vartheta"] &
\end{tikzcd}$$

Proof. Due to the definition of the functors $\mathbf{Int}_\Sigma : \mathsf{Cxt}_S^{op} \to \mathsf{CAT}$ and $\mathbf{Stm}_\Sigma : \mathsf{Cxt}_S \to \mathsf{Set}$, we obtain the commutative diagram, above on the right, thus the satisfaction condition follows immediately from Definition 12 (Satisfaction Relation). □

Summarizing all definitions and results, we obtain for each signature Σ the **Institution of Σ-Statements in Context** $\mathcal{FL}_\Sigma = (\mathsf{Cxt}_S, \mathbf{Stm}_\Sigma, \mathbf{Int}_\Sigma, \models^\Sigma)$.

Remark 16 (Empty Sorts). *The fact that the rules for unsorted first-order logic are not sound for many-sorted first-order logic is caused by the freedom that some (or even all) components A_s of the carrier S-set A of a Σ-structure $\mathcal{A}$ can, in principle, be empty. This freedom is essential for applications of formal specifications as discussed, for example, in [26].*

By introducing the conceptual level of contexts we are able to address and potentially deal with this issue. As said in the introduction, we will not present in this paper a comprehensive exposition of deduction but only discuss some aspects of deduction.

Deduction in Logics of Statements in Context means, in the first place, to deduce new statements in contexts from given statements in contexts by means of rules. The problem, caused by potentially empty sorts, can be identified by the observation that the equivalence in Remark 15 vanishes if there is no Σ-interpretation $(\iota,\mathcal{A})$ of the context K in the Σ-structure $\mathcal{A}$ at all.

A rule, stating that we can deduce from the validity of $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ in the empty context $\mathbf{0}_S$ the validity of $(\mathbf{0}_S, Ex, !_K)$ in any context K, will be always sound since $(\mathbf{0}_S, Ex, !_K)$ is simply the translation of $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ along the context morphism $!_K : \mathbf{0}_S \to K$ (see rule "Abstraction" in [19]). However, a rule stating that we can deduce from the validity of $(\mathbf{0}_S, Ex, !_K)$ in a certain context K the validity of $(\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ in the empty context $\mathbf{0}_S$ will be only sound if there is, at least, one interpretation $(\iota : K \to A, \mathcal{A})$ of context K in all Σ-structure $\mathcal{A}$! This requirement can be only fulfilled if $T_\Sigma(\mathbf{0}_S)_s \neq \emptyset$ whenever $K_s \neq \emptyset$ for all $s \in S$ (see rule "Concretion" in [19]).

5.2. The Indexed First-Order Logic of Statements in Context

5.2.1. Institution Comorphisms

For any signature morphism $\varphi : \Sigma \to \Sigma'$ between signatures $\Sigma = (S, P, F, ar, in, out)$ and $\Sigma' = (S', P', F', ar', in', out')$, as defined in Definition 7, we consider the institutions $\mathcal{FL}_\Sigma = (\mathsf{Cxt}_S, \mathbf{Stm}_\Sigma, \mathbf{Int}_\Sigma, \models^\Sigma)$ and $\mathcal{FL}_{\Sigma'} = (\mathsf{Cxt}_{S'}, \mathbf{Stm}_{\Sigma'}, \mathbf{Int}_{\Sigma'}, \models^{\Sigma'})$ as described in the last subsection. We will construct an **institution comorphism** $(\Phi_\varphi, \alpha^\varphi, \beta^\varphi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma'}$ consisting of (see [17] or [27] for a general definition of institution comorphisms)

$$\begin{tikzcd}
\mathsf{Cxt}_S \arrow[r, "\mathbf{Stm}_\Sigma"] \arrow[d, "\Phi_\varphi"'] & \mathsf{Set} \\
\mathsf{Cxt}_{S'} \arrow[ur, "\mathbf{Stm}_{\Sigma'}"'] \arrow[ur, phantom, bend left=40, "\alpha^\varphi \Downarrow"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Cxt}_S^{op} \arrow[r, "\mathbf{Int}_\Sigma"] \arrow[d, "\Phi_\varphi^{op}"'] & \mathsf{CAT} \\
\mathsf{Cxt}_{S'}^{op} \arrow[ur, "\mathbf{Int}_{\Sigma'}"'] \arrow[ur, phantom, bend left=40, "\beta^\varphi \Uparrow"] &
\end{tikzcd}$$

1. a functor $\Phi_\varphi : \mathsf{Cxt}_S \to \mathsf{Cxt}_{S'}$,
2. a natural transformation $\alpha^\varphi : \mathbf{Stm}_\Sigma \Rightarrow \Phi_\varphi ; \mathbf{Stm}_{\Sigma'}$, and
3. a natural transformation $\beta^\varphi : \Phi_\varphi^{op} ; \mathbf{Int}_{\Sigma'} \Rightarrow \mathbf{Int}_\Sigma$

such that the following **satisfaction condition for comorphisms** holds:

$$\beta^{\varphi}_K(\iota',\mathcal{A}') \models^{\Sigma}_K (X,Ex,\gamma) \quad \text{if, and only if,} \quad (\iota',\mathcal{A}') \models^{\Sigma'}_{\Phi_\varphi(K)} \alpha^{\varphi}_K(X,Ex,\gamma) \tag{19}$$

for any context $K \in |\mathsf{Cxt}_S|$, any Σ'-interpretation $(\iota',\mathcal{A}')$ of $\Phi_\varphi(K) \in |\mathsf{Cxt}'_S|$ and any Σ-statement (X,Ex,γ) in context K.

By definition we have $\mathsf{Cxt}_S = \mathsf{Set}^S_d$ and $\mathsf{Cxt}_{S'} = \mathsf{Set}^{S'}_d$ thus we can simply set

$$\Phi_\varphi := \mathbf{S}^d_\varphi : \mathsf{Set}^S_d \to \mathsf{Set}^{S'}_d. \tag{20}$$

Note, that there is no right adjoint for Φ_φ thus we can not equivalently describe the institution comorphism $(\Phi_\varphi, \alpha^\varphi, \beta^\varphi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma'}$ by means of an institution morphism!

Translation of Statements along Signature Morphisms

For any context K in $\mathsf{Cxt}_S = \mathsf{Set}^S_d$ we define the map $\alpha^\varphi_K : Stm_\Sigma(K) \to Stm_{\Sigma'}(\mathbf{S}^d_\varphi(K))$ by means of the translation of expressions defined in Section 4.2.3:

$$\alpha^\varphi_K(X,Ex,\gamma) := (\mathbf{S}^d_\varphi(X), \overline{\varphi}_X(Ex), \mathbf{S}^d_\varphi(\gamma)) \quad \text{for any } (X,Ex,\gamma) \in Stm_\Sigma(K). \tag{21}$$

The assignments $K \mapsto \alpha^\varphi_K$ constitute a natural transformation $\alpha^\varphi : \mathbf{Stm}_\Sigma \Rightarrow \Phi_\varphi ; \mathbf{Stm}_{\Sigma'}$ since we defined the translation of statements along context morphisms by simple post-composition and since $\Phi_\varphi = \mathbf{S}^d_\varphi : \mathsf{Set}^S_d \to \mathsf{Set}^{S'}_d$ is a functor.

$$\begin{tikzcd}
K \arrow[d, "\mu"'] & Stm_\Sigma(K) \arrow[r, "\alpha^\varphi_K"] \arrow[d, "(\_;\mu)"'] \arrow[dr, phantom, "="] & Stm_{\Sigma'}(\mathbf{S}^d_\varphi(K)) \arrow[d, "(\_;\mathbf{S}^d_\varphi(\mu))"] \\
G & Stm_\Sigma(G) \arrow[r, "\alpha^\varphi_G"] & Stm_{\Sigma'}(\mathbf{S}^d_\varphi(G))
\end{tikzcd}$$

Translation of Interpretations along Signature Morphisms

For any context K in $\mathsf{Cxt}_S = \mathsf{Set}^S_d$ we define the functor $\beta^\varphi_K : \mathsf{Int}_{\Sigma'}(\mathbf{S}^d_\varphi(K)) \to \mathsf{Int}_\Sigma(K)$ by means of the bijections in (7) and the reduct of structures defined in Section 4.2.1:

$$\beta^\varphi_K(\iota',\mathcal{A}') := (\mathbf{B}_\varphi(\iota'), \mathcal{A}'\upharpoonright_\varphi) \quad \text{for any } \Sigma'\text{-interpretation } (\iota',\mathcal{A}') \in \mathsf{Int}_{\Sigma'}(\mathbf{S}^d_\varphi(K)). \tag{22}$$

The assignments $K \mapsto \beta^\varphi_K$ constitute a natural transformation $\beta^\varphi : \Phi^{op}_\varphi ; \mathbf{Int}_{\Sigma'} \Rightarrow \mathbf{Int}_\Sigma$ since we defined the translation of interpretations along context morphisms in (15) by simple pre-composition and due to (10) (see the diagram below on the left).

$$\begin{tikzcd}
K \arrow[d, "\mu"'] & \mathsf{Int}_{\Sigma'}(\mathbf{S}^d_\varphi(K)) \arrow[r, "\beta^\varphi_K"] \arrow[dr, phantom, "="] & \mathsf{Int}_\Sigma(K) \\
G & \mathsf{Int}_{\Sigma'}(\mathbf{S}^d_\varphi(G)) \arrow[r, "\beta^\varphi_G"] \arrow[u, "\mathbf{Int}_\Sigma(\mathbf{S}^d_\varphi(\mu))"] & \mathsf{Int}_\Sigma(G) \arrow[u, "\mathbf{Int}_\Sigma(\mu)"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Int}_{\Sigma'}(\mathbf{S}^d_\varphi(K)) \arrow[r, "\beta^\varphi_K"] \arrow[d, "\Pi_{\mathbf{S}^d_\varphi(K)}"'] \arrow[dr, phantom, "="] & \mathsf{Int}_\Sigma(K) \arrow[d, "\Pi_K"] \\
\mathsf{Str}(\Sigma') \arrow[r, "\mathbf{Str}(\varphi)=(\_\upharpoonright_\varphi)"] & \mathsf{Str}(\Sigma)
\end{tikzcd}$$

Supplementing (16), we observe that β^φ_K is defined in (22) in such a way that

$$\beta^\varphi_K ; \Pi_K = \Pi_{\mathbf{S}^d_\varphi(K)} ; \mathbf{Str}(\varphi) \quad \text{(see the diagram above on the right).} \tag{23}$$

Satisfaction Condition for Comorphisms

Let be given a context $K \in |\mathsf{Cxt}_S|$, a Σ'-interpretation $(\iota', \mathcal{A}')$ of $\Phi_\varphi(K) \in |\mathsf{Cxt}_{S'}|$ and a Σ-statement (X, Ex, γ) in context K.

$$\begin{tikzcd}
 & \mathbf{S}^d_\varphi(K) \arrow[dl, "\iota'"'] & \\
\mathcal{A}' & & \mathbf{S}^d_\varphi(X) \rhd \overline{\varphi}_X(Ex) \arrow[ul, "\mathbf{S}^d_\varphi(\gamma)"'] \arrow[ll, "\mathbf{S}^d_\varphi(\gamma);\iota'"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & K \arrow[dl, "\mathbf{B}_\varphi(\iota')"'] & \\
\mathbf{R}_\varphi(\mathcal{A}') & & X \rhd Ex \arrow[ul, "\gamma"'] \arrow[ll, "\gamma;\mathbf{B}_\varphi(\iota')"']
\end{tikzcd}$$

First, due to the definition of β^φ_K in (22), the statement $\beta^\varphi_K(\iota', \mathcal{A}') \models^\Sigma_K (X, Ex, \gamma)$ is equivalent to the statement $(\mathbf{B}_\varphi(\iota'), \mathcal{A}' \upharpoonright_\varphi) \models^\Sigma_K (X, Ex, \gamma)$ while this statement is equivalent to $\mathbf{B}_\varphi(\mathbf{S}^d_\varphi(\gamma); \iota') = \gamma; \mathbf{B}_\varphi(\iota') \models^\Sigma_{\mathcal{A}' \upharpoonright_\varphi} X \rhd Ex$ according to (10) and Definition 12.

Second, due to the definition of Φ_φ in (20) and the definition of α^φ_K in (21) the statement $(\iota', \mathcal{A}') \models^{\Sigma'}_{\Phi_\varphi(K)} \alpha^\varphi_K(X, Ex, \gamma)$ is equivalent to $(\iota', \mathcal{A}') \models^{\Sigma'}_{\mathbf{S}^d_\varphi(K)} (\mathbf{S}^d_\varphi(X), \overline{\varphi}_X(Ex), \mathbf{S}^d_\varphi(\gamma))$ while this statement is equivalent to $\mathbf{S}^d_\varphi(\gamma); \iota' \models^{\Sigma'}_{\mathcal{A}'} \mathbf{S}^d_\varphi(X) \rhd \overline{\varphi}_X(Ex)$ by Definition 12.

In such a way, the required satisfaction condition for comorphisms in (19) exactly reflects the general satisfaction condition for expressions and assignments in (12)!

5.2.2. Composition of Comorphisms

For any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ the composition of the institution comorphisms $(\Phi_\varphi, \alpha^\varphi, \beta^\varphi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma'}$ and $(\Phi_\psi, \alpha^\psi, \beta^\psi) : \mathcal{FL}_{\Sigma'} \to \mathcal{FL}_{\Sigma''}$ gives rise to an institution comorphism $(\Phi_\varphi, \alpha^\varphi, \beta^\varphi); (\Phi_\psi, \alpha^\psi, \beta^\psi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma''}$ given by (compare [17])

$$\begin{tikzcd}
\mathsf{Cxt}_S \arrow[d, "\Phi_\varphi"'] \arrow[drr, "\mathbf{Stm}_\Sigma"] & {} \arrow[d, Rightarrow, "\alpha^\varphi"'] & \\
\mathsf{Cxt}_{S'} \arrow[d, "\Phi_\psi"'] \arrow[rr, "\mathbf{Stm}_{\Sigma'}"] & {} \arrow[d, Rightarrow, "\alpha^\psi"'] & \mathsf{Set} \\
\mathsf{Cxt}_{S''} \arrow[urr, "\mathbf{Stm}_{\Sigma''}"'] & {} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Cxt}^{op}_S \arrow[d, "\Phi^{op}_\varphi"'] \arrow[drr, "\mathbf{Int}_\Sigma"] & {} & \\
\mathsf{Cxt}^{op}_{S'} \arrow[d, "\Phi^{op}_\psi"'] \arrow[rr, "\mathbf{Int}_{\Sigma'}"] & {} \arrow[u, Rightarrow, "\beta^\varphi"] & \mathsf{CAT} \\
\mathsf{Cxt}^{op}_{S''} \arrow[urr, "\mathbf{Int}_{\Sigma''}"'] & {} \arrow[u, Rightarrow, "\beta^\psi"] &
\end{tikzcd}$$

1. the functor $\Phi_\varphi; \Phi_\psi : \mathsf{Cxt}_S \to \mathsf{Cxt}_{S''}$,
2. the natural transformation $\alpha^\varphi; (\Phi_\varphi . \alpha^\psi) : \mathbf{Stm}_\Sigma \Rightarrow (\Phi_\varphi; \Phi_\psi); \mathbf{Stm}_{\Sigma'}$, and
3. the natural transformation $(\Phi^{op}_\varphi . \beta^\psi); \beta^\varphi : (\Phi_\varphi; \Phi_\psi)^{op}; \mathbf{Int}_{\Sigma''} \Rightarrow \mathbf{Int}_\Sigma$.

The definition of the institution comorphisms $(\Phi_\varphi, \alpha^\varphi, \beta^\varphi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma'}$ in Section 5.2.1 ensures that their composition can be represented by means of the composition of the underlying signature morphisms.

Lemma 3 (Composition of Institution Comorphisms)**.**
For any signature morphisms $\varphi : \Sigma \to \Sigma'$ and $\psi : \Sigma' \to \Sigma''$ we have

$$(\Phi_{\varphi;\psi}, \alpha^{\varphi;\psi}, \beta^{\varphi;\psi}) = (\Phi_\varphi; \Phi_\psi, \alpha^\varphi; (\Phi_\varphi . \alpha^\psi), (\Phi^{op}_\varphi . \beta^\psi); \beta^\varphi) = (\Phi_\varphi, \alpha^\varphi, \beta^\varphi); (\Phi_\psi, \alpha^\psi, \beta^\psi).$$

Proof. Since $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$ is a functor, we trivially have $\Phi_{\varphi;\psi} = \Phi_\varphi; \Phi_\psi$ due to (20). Since $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$ is a functor, $\alpha^{\varphi;\psi} = \alpha^\varphi; (\Phi_\varphi . \alpha^\psi)$ is an immediate consequence of Lemma 2 due to the definitions in (20) and (21). Since $\mathbf{S}^d : \mathsf{Sort} \to \mathsf{CAT}$ and $\mathbf{Str} : \mathsf{Sig}^{op} \to \mathsf{CAT}$ are functors, the fact that adjunctions compose, finally ensures $\beta^{\varphi;\psi} = (\Phi^{op}_\varphi . \beta^\psi); \beta^\varphi$ due to the definitions in (20) and (22). □

That the assignments $\Sigma \mapsto \mathcal{FL}_\Sigma = (\mathsf{Cxt}_S, \mathbf{Stm}_\Sigma, \mathbf{Int}_\Sigma, \models^\Sigma)$ and $(\varphi : \Sigma \to \Sigma') \mapsto ((\Phi_\varphi, \alpha^\varphi, \beta^\varphi) : \mathcal{FL}_\Sigma \to \mathcal{FL}_{\Sigma'})$ define a functor (also called an *indexed comorphism-based institution* [17])

$$\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns \tag{24}$$

from the category Sig of many-sorted signatures into the category $co\mathbb{I}ns$ of institutions and comorphisms, is an immediate consequence of Lemma 3. We call $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ the **Indexed First-Order Logic of Statements in Context**.

5.3. The Fibred First-Order Logic of Statements in Context

5.3.1. A Grothendieck Institution

In contrast to [17,18], we ended up with a covariant functor $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ and not a contravariant functor as an *indexed comorphism-based institution* [17]. As mentioned already in Remark 10, there are actually four possibilities to transform an indexed category into a split (op)fibration not only the standard Grothendieck construction usually presented in the literature [8]. Correspondingly, there are, at least, four possibilities to transform an indexed institution into a Grothendieck institution. We will not present the general construction of Grothendieck institutions for covariant indexed comorphism-based institutions, but define directly a Grothendieck institution for $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$.

The Grothendieck institution $\mathcal{FL}^\sharp = (\mathsf{Sig}^\sharp, \mathbf{Stm}^\sharp, \mathbf{Int}^\sharp, \models^\sharp)$ for $\mathcal{FL} : \mathsf{Sig} \to co\mathbb{I}ns$ is constructed as follows:

1. The category $\mathsf{Sig}^\sharp$ has as
   - objects pairs (Σ, K) with $\Sigma = (S, P, F, ar, in, out) \in |\mathsf{Sig}|$ and $K \in |\mathsf{Cxt}_\mathsf{S}|$, and as
   - morphisms pairs $(\varphi, \mu') : (\Sigma, K) \to (\Sigma', G')$ of a signature morphism $\varphi : \Sigma \to \Sigma'$ and a context morphism $\mu' : \mathbf{S}^d_\varphi(K) \to G'$ where $\Phi_\varphi = \mathbf{S}^d_\varphi : \mathsf{Set}^S_d \to \mathsf{Set}^{S'}_d$.
   - The composition of $(\varphi, \mu') : (\Sigma, K) \to (\Sigma', G')$ and $(\psi, \mu'') : (\Sigma', G') \to (\Sigma'', H'')$ is the morphism $(\varphi;\psi, \mathbf{S}^d_\psi(\mu');\mu'') : (\Sigma, K) \to (\Sigma'', H'')$.
2. The sentence functor $\mathbf{Stm}^\sharp : \mathsf{Sig}^\sharp \to \mathsf{Set}$ is given by
   - $\mathbf{Stm}^\sharp(\Sigma, K) := Stm_\Sigma(K)$ for each object (Σ, K) in $\mathsf{Sig}^\sharp$, and
   - $\mathbf{Stm}^\sharp(\varphi, \mu') := \alpha^\varphi_K; \mathbf{Stm}_{\Sigma'}(\mu')$ for each morphism $(\varphi, \mu') : (\Sigma, K) \to (\Sigma', G')$, i.e., for all $(X, Ex, \gamma) \in \mathbf{Stm}^\sharp(\Sigma, K) = Stm_\Sigma(K)$ we have $\mathbf{Stm}^\sharp(\varphi, \mu')(X, Ex, \gamma) = (\mathbf{S}^d_\varphi(X), \overline{\varphi}_X(Ex), \mathbf{S}^d_\varphi(\gamma);\mu') \in \mathbf{Stm}^\sharp(\Sigma', G') = Stm_{\Sigma'}(G')$.
3. The model functor $\mathbf{Int}^\sharp : (\mathsf{Sig}^\sharp)^{op} \to \mathsf{CAT}$ is given by
   - $\mathbf{Int}^\sharp(\Sigma, K) := \mathsf{Int}_\Sigma(K)$ for each object (Σ, K) in $\mathsf{Sig}^\sharp$, and
   - $\mathbf{Int}^\sharp(\varphi, \mu') := \mathbf{Int}_{\Sigma'}(\mu'); \beta^\varphi_K$ for each morphism $(\varphi, \mu') : (\Sigma, K) \to (\Sigma', G')$, i.e., for all $(\iota', \mathcal{A}') \in \mathbf{Int}^\sharp(\Sigma', G') = \mathsf{Int}_{\Sigma'}(G')$ we have $\mathbf{Int}^\sharp(\varphi, \mu')(\iota', \mathcal{A}') = (\mathbf{B}_\varphi(\mu';\iota'), \mathcal{A}' \upharpoonright_\varphi) \in \mathbf{Int}^\sharp(\Sigma, K) = \mathsf{Int}_\Sigma(K)$.
4. The satisfaction relation $\models^\sharp$ is defined by $(\iota, \mathcal{A}) \models^\sharp_{(\Sigma,K)} (X, Ex, \gamma)$ if, and only if, $(\iota, \mathcal{A}) \models^\Sigma_K (X, Ex, \gamma)$ for each $\Sigma \in |\mathsf{Sig}|$, $K \in |\mathsf{Cxt}_\mathsf{S}|$, $(\iota, \mathcal{A}) \in \mathbf{Int}^\sharp(\Sigma, K) = \mathsf{Int}_\Sigma(K)$ and $(X, Ex, \gamma) \in \mathbf{Stm}^\sharp(\Sigma, K) = Stm_\Sigma(K)$.

By routine calculations, we can check that the definition of $\mathcal{FL}^\sharp$ transforms the satisfaction condition for comorphisms in (19) to the satisfaction condition for $\mathcal{FL}^\sharp$:

$$\mathbf{Int}^\sharp(\varphi, \mu')(\iota', \mathcal{A}') \models^\sharp_{(\Sigma,K)} (X, Ex, \gamma) \quad \text{iff} \quad (\iota', \mathcal{A}') \models^\sharp_{(\Sigma',G')} \mathbf{Stm}^\sharp(\varphi, \mu')(X, Ex, \gamma) \tag{25}$$

for any morphism $(\varphi, \mu') : (\Sigma, K) \to (\Sigma', G')$ in $\mathsf{Sig}^\sharp$, any statement $(X, Ex, \gamma) \in \mathbf{Stm}^\sharp(\Sigma, K) = Stm_\Sigma(K)$ and any interpretation $(\iota', \mathcal{A}') \in |\mathbf{Int}^\sharp(\Sigma', \mathsf{G}')| = |\mathsf{Int}_{\Sigma'}(\mathsf{G}')|$.

5.3.2. $\mathcal{FL}^\sharp$ as an Extension of $\mathcal{FOL}$

We validate that we indeed achieved our objective to construct an institution extending the traditional institution $\mathcal{FOL} = (\mathsf{Sig}, \mathbf{cFE}, \mathbf{Str}, \models)$ of first-order logic, as described in Section 4.2.5, by arbitrary first-order "open formulas". We show that $\mathcal{FOL}$ can be fully embedded into $\mathcal{FL}^\sharp = (\mathsf{Sig}^\sharp, \mathbf{Stm}^\sharp, \mathbf{Int}^\sharp, \models^\sharp)$. Varying the terminology in [17], we may say that $\mathcal{FOL}$ is a "full sub-institution of $\mathcal{FL}^\sharp$ up to renaming".

First, we observe that the assignments $\Sigma \mapsto (\Sigma, \mathbf{0}_S)$ and $(\varphi : \Sigma \to \Sigma') \mapsto ((\varphi, id_{\mathbf{0}_{S'}}) : (\Sigma, \mathbf{0}_S) \to (\Sigma', \mathbf{0}_{S'}))$ define a full embedding $\mathbf{Em} : \mathsf{Sig} \to \mathsf{Sig}^\sharp$. Keep in mind $\mathbf{S}^d_\varphi(\mathbf{0}_S) = \mathbf{0}_{S'}$.

$$\begin{tikzcd}
\mathsf{Sig} \arrow[r, "\mathbf{cFE}"] \arrow[d, "\mathbf{Em}"'] \arrow[dr, phantom, "\nu^{sn}\Downarrow"] & \mathsf{Set} \\
\mathsf{Sig}^\sharp \arrow[ur, "\mathbf{Stm}^\sharp"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Sig}^{op} \arrow[r, "\mathbf{Str}"] \arrow[d, "\mathbf{Em}^{op}"'] \arrow[dr, phantom, "\nu^{md}\Downarrow"] & \mathsf{CAT} \\
(\mathsf{Sig}^\sharp)^{op} \arrow[ur, "\mathbf{Int}^\sharp"'] &
\end{tikzcd}$$

Second, there is a natural isomorphism $\nu^{sn} : \mathbf{cFE} \Rightarrow \mathbf{Em}; \mathbf{Stm}^\sharp$ with maps

$$\nu^{sn}_\Sigma : FE_\Sigma(\mathbf{0}_S) \longrightarrow \mathbf{Stm}^\sharp(\Sigma, \mathbf{0}_S) = \mathbf{Stm}_\Sigma(\mathbf{0}_S) \quad \text{for all } \Sigma \in |\mathsf{Sig}|$$

defined by $\nu^{sn}_\Sigma(Ex) := (\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ for all $Ex \in FE_\Sigma(\mathbf{0}_S)$. The map ν^{sn}_Σ is bijective since $id_{\mathbf{0}_S}$ is the only S-map with codomain $\mathbf{0}_S$! The naturality of ν^{sn} immediately follows from the definition of $\mathbf{Stm}^\sharp$ in Section 5.3.1 and the definition of α^φ in (21).

Third, there is a natural isomorphism $\nu^{md} : \mathbf{Str} \Rightarrow \mathbf{Em}^{op}; \mathbf{Int}^\sharp$ with isomorphisms

$$\nu^{md}_\Sigma := \Pi^{-1}_{\mathbf{0}_S} : \mathsf{Str}(\Sigma) \longrightarrow \mathbf{Int}^\sharp(\Sigma, \mathbf{0}_S) = \mathbf{Int}_\Sigma(\mathbf{0}_S) \quad \text{for all } \Sigma \in |\mathsf{Sig}|$$

according to Remark 14. The naturality of ν^{md} immediately follows from the definition of $\mathbf{Int}^\sharp$ in Section 5.3.1 and (23).

Fourth, the statement $(!_A, \mathcal{A}) \models^\sharp_{(\Sigma, id_{\mathbf{0}_S})} (\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$ for any signature Σ and any Σ-structure $\mathcal{A}$ is equivalent to the statement $(!_A, \mathcal{A}) \models^\Sigma_{\mathbf{0}_S} (\mathbf{0}_S, Ex, id_{\mathbf{0}_S})$, due to the definition of $\models^\sharp$ in Section 5.3.1, and Definition 12 entails that this statement is, in turn, equivalent to the statement $id_{\mathbf{0}_S}; !_A = !_A \models^\Sigma_{\mathcal{A}} \mathbf{0}_S \triangleright Ex$. This is, however, exactly how we defined the satisfaction relation $\mathcal{A} \models^\Sigma \mathbf{0}_S \triangleright Ex$ for $\mathcal{FOL}$ in Section 4.2.5. In other words, the satisfaction condition in $\mathcal{FOL}$ is a restriction of the satisfaction condition in $\mathcal{FL}^\sharp$!

5.3.3. A Fibred Institution

In addition of $\mathcal{FOL}$ being a full sub-institution of $\mathcal{FL}^\sharp$, the construction of $\mathcal{FL}^\sharp$ also entails that $\mathcal{FL}^\sharp$ is fibred over $\mathcal{FOL}$. The concept of *fibred institutions* in [17] is based on fibrations while we meet here opfibrations instead. Moreover, the concept of fibred institution in [17] only relies on a fibration for the category of signatures. We will not coin a general concept of an institution fibred over another institution, but simply describe the strong relationship between $\mathcal{FL}^\sharp$ and $\mathcal{FOL}$ in terms of opfibrations.

The assignments $(\Sigma, K) \mapsto \Sigma$ and $(\varphi, \mu') \mapsto \varphi$ define a functor $\mathbf{P} : \mathsf{Sig}^\sharp \to \mathsf{Sig}$ right-adjoint to the full embedding $\mathbf{Em} : \mathsf{Sig} \to \mathsf{Sig}^\sharp$ with $\mathbf{Em}; \mathbf{P} = id_{\mathsf{Sig}}$. Moreover, $\mathbf{P} : \mathsf{Sig}^\sharp \to \mathsf{Sig}$ is a split opfibration with the opcartesian arrow $(\varphi, id_{\mathbf{S}^d_\varphi(K)}) : (\Sigma, K) \to (\Sigma', \mathbf{S}^d_\varphi(K))$ for a signature morphisms $\varphi : \Sigma \to \Sigma'$ and an object (Σ, K) in $\mathsf{Sig}^\sharp$.

As discussed in Remark 14, the projection functors $\Pi_K : \mathsf{Int}_\Sigma(K) \to \mathsf{Str}(\Sigma)$ are split opfibrations. The definition of $\mathbf{Int}^\sharp : (\mathsf{Sig}^\sharp)^{op} \to \mathsf{CAT}$ in Section 5.3.1 and the equations (16), (23) ensure that the family of the obfibrations $\Pi_K : \mathsf{Int}_\Sigma(K) \to \mathsf{Str}(\Sigma)$ establishes a natural transformation $\Pi : \mathbf{Int}^\sharp \Rightarrow \mathbf{P}^{op}; \mathbf{Str}$.

$$\begin{tikzcd}
\mathsf{Sig}^\sharp \arrow[rr, "\mathbf{Stm}^\sharp"] \arrow[d, "\mathbf{P}"'] & {} \arrow[d, phantom, "\Upsilon\Downarrow"] & \mathsf{Set} \\
\mathsf{Sig} \arrow[rru, "\mathbf{V}"'] & {} &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
(\mathsf{Sig}^\sharp)^{op} \arrow[rr, "\mathbf{Int}^\sharp"] \arrow[d, "\mathbf{P}^{op}"'] & {} \arrow[d, phantom, "\Pi\Downarrow"] & \mathsf{CAT} \\
\mathsf{Sig}^{op} \arrow[rru, "\mathbf{Str}"'] & {} &
\end{tikzcd}$$

On the syntactic side, we do have another functor $\mathbf{V} : \mathsf{Sig} \to \mathsf{Set}$ given by the assignments $\Sigma = (S, P, F, ar, in, out) \mapsto Var_S$ and $\varphi = (\varphi_{st}, \varphi_{rl}, \varphi_{fn}) \mapsto (\mathbf{S}^d_{\varphi_{st}} : Var_S \to Var_{S'})$.

For any object (Σ, K) in $\mathsf{Sig}^\sharp$ there is a map $\Upsilon_K : Stm_\Sigma(K) \to Var_S$ given by the assignments $(X, Ex, \gamma) \mapsto X$. The definition of $\mathbf{Stm}^\sharp : \mathsf{Sig}^\sharp \to \mathsf{Set}$ in Section 5.3.1 and the definitions in (14), (21) ensure that the family of the maps $\Upsilon_K : Stm_\Sigma(K) \to Var_S$ establishes a natural transformation $\Upsilon : \mathbf{Stm}^\sharp \Rightarrow \mathbf{P}; \mathbf{V}$.

We call $\mathcal{FL}^\sharp = (\mathsf{Sig}^\sharp, \mathbf{Stm}^\sharp, \mathbf{Int}^\sharp, \models^\sharp)$ together with the opfibration $\mathbf{P} : \mathsf{Sig}^\sharp \to \mathsf{Sig}$ and the natural transformations $\Pi : \mathbf{Int}^\sharp \Rightarrow \mathbf{P}^{op}; \mathbf{Str}$, $\Upsilon : \mathbf{Stm}^\sharp \Rightarrow \mathbf{P}; \mathbf{V}$ the **Fibred First-Order Logic of Statements in Context**.

Remark 17 (Applications)**.** *Fortunately, we do have now two equivalent ways at our disposal to deal with fully-fleshed first-order "open formulas". It is, however, nearly impossible to state a priori which way will be most appropriate in applications. We guess that it will be, at the end, most convenient in many applications to utilize both ways in a well-defined interplay (compare [28], for example).*

6. Substitutions

Substitutions are not relevant for utilizing a logic as a specification formalism. As also mentioned in [17], substitutions are, however, an important logical device for deduction. In this section we investigate substitutions and develop corresponding extensions of constructions and results in Sections 3 and 4 necessary and sufficient for deduction.

6.1. Categories of Substitutions

The simple, natural idea is to extend the sets Var_S of S-sets of variables by substitutions. For any signature $\Sigma = (S, P, F, ar, in, out)$ the small category Var_Σ is defined as follows:

Objects: are all S-sets X of variables, i.e., $|\mathsf{Var}_\Sigma| := Var_S$;
Morphisms: $\boldsymbol{t} : X \to Y$ in Var_Σ are given by Σ-substitutions $\boldsymbol{t} : X \to T_\Sigma(Y)$;
Identities: on S-sets X are given by the canonical S-maps η_X: $X \to T_\Sigma(X)$; and the
Composition: $\boldsymbol{t};^\Sigma \boldsymbol{r} : X \to Z$ of two morphisms $\boldsymbol{t} : X \to Y$, $\boldsymbol{r} : Y \to Z$ in Var_Σ is given by the Σ-substitution $\boldsymbol{t};\boldsymbol{r}^* : X \to T_\Sigma(Z)$ where $\boldsymbol{r}^* : T_\Sigma(Y) \to T_\Sigma(Z)$ is the inductive extension of $\boldsymbol{r} : Y \to T_\Sigma(Z)$ such that

$$\boldsymbol{r} = \eta_Y;\boldsymbol{r}^* . \qquad \begin{tikzcd}
Y \arrow[r, "\eta_Y"] \arrow[rd, "\boldsymbol{r}"'] & T_\Sigma(Y) \arrow[d, "\boldsymbol{r}^*"] \\
& T_\Sigma(Z)
\end{tikzcd} \tag{26}$$

Note, that $\boldsymbol{r}^* : T_\Sigma(Y) \to T_\Sigma(Z)$ describes nothing but the application of the Σ-substitution $\boldsymbol{r} : Y \to T_\Sigma(Z)$ to all Σ-terms over Y! We can inductively prove that

$$\eta_X^* = id_{T_\Sigma(X)} \quad \text{and} \quad (\boldsymbol{t};\boldsymbol{r}^*)^* = \boldsymbol{t}^*;\boldsymbol{r}^* \tag{27}$$

thus the composition of substitutions is associative, and we get indeed a category.

Remark 18 (Lawvere Theories)**.** *The tentative reader may have realized that the categories* Var_Σ *are nothing but so-called* syntactic Lawvere theories *[21,22,29]. We will, however, not walk further into the realm of "Categorical Algebra" and "Functorial Semantics". We will neither utilize the fact that the categories* Var_Σ *are* *finite product categories nor reconstruct* $\mathsf{Str}(\Sigma)$ *as a subcategory of the functor category* $[\mathsf{Var}_\Sigma \to \mathsf{Set}]$ *in case of signatures* Σ *without predicate symbols!*

Remark 19 (Kleisli Morphisms)**.** *It is a common practice to describe substitutions as morphisms of the Kleisli category of a "term algebra adjunction". In such a way, we would get the extension of assignments in (3) and the application of substitutions in (26) as well as their compatibility "for free". See the discussion of "internalization of terms" and substitution calculi in [15].*

To stay closer to traditional presentations of logics, we rely instead in this paper on explicit inductive definitions (located on the meta-level). Another reason is, that some of the necessary constructions and results, as the translation of terms and Lemma 1 for example, can not be obtained by means of term algebra adjunctions (see Remark 11)!

For a signature morphism $\varphi : \Sigma \to \Sigma'$ we can extend the map $\mathbf{S}^d_\varphi : Var_S \to Var_{S'}$ in Section 4.2.2 to a functor $\mathbf{L}_\varphi : \mathsf{Var}_\Sigma \to \mathsf{Var}_{\Sigma'}$ defined by $\mathbf{L}_\varphi(\boldsymbol{t}) := \mathbf{S}^d_\varphi(\boldsymbol{t});\overline{\varphi}_Y$ for all substitutions $\boldsymbol{t} : X \to T_\Sigma(Y)$ where $\overline{\varphi} : T_\Sigma\,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{S}^d_\varphi\,;T_{\Sigma'} : Var_S \to \mathsf{Set}^{S'}_d$ is the natural transformation defined in Section 4.2.2 and describing the translation of Σ-terms into Σ'-terms.

$$\begin{tikzcd}
X \arrow[d, "\boldsymbol{t}"'] \\
T_\Sigma(Y)
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{S}^d_\varphi(X) \arrow[d, "\mathbf{S}^d_\varphi(\boldsymbol{t})"'] \arrow[dr, "\mathbf{L}_\varphi(\boldsymbol{t})"] & \\
\mathbf{S}^d_\varphi(T_\Sigma(Y)) \arrow[r, "\overline{\varphi}_Y"] & T_{\Sigma'}(\mathbf{S}^d_\varphi(Y))
\end{tikzcd}$$

Lemma 1 ensures $\mathbf{L}_{\varphi;\psi} = \mathbf{L}_\varphi;\mathbf{L}_\psi$ for any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ thus the assignments $\Sigma \mapsto \mathsf{Var}_\Sigma$ and $\varphi \mapsto \mathbf{L}_\varphi$ define a functor $\mathbf{L} : \mathsf{Sig} \to \mathsf{Cat}$.

6.2. Translation of Terms and Substitution Application

We can also extend the maps $T_\Sigma : Var_S \to \mathsf{Set}^S_d$ to functors $\mathbf{T}_\Sigma : \mathsf{Var}_\Sigma \to \mathsf{Set}^S_d$ with $\mathbf{T}_\Sigma(X) := T_\Sigma(X)$ for all S-sets X of variables and $\mathbf{T}_\Sigma(\boldsymbol{t}) := \boldsymbol{t}^*$, due to (26), for all Σ-substitutions $\boldsymbol{t} : X \to T_\Sigma(Y)$. Functoriality is ensured by (27).

To show that the family of S'-maps $\overline{\varphi}_X : \mathbf{S}^d_\varphi(T_\Sigma(X)) \to T_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ also constitutes a natural transformation $\overline{\varphi} : \mathbf{T}_\Sigma\,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{L}_\varphi\,;\mathbf{T}_{\Sigma'} : \mathsf{Var}_\Sigma \to \mathsf{Set}^{S'}_d$, we have to prove that for any Σ-substitution $\boldsymbol{t} : X \to T_\Sigma(Y)$ the square of S'-maps on the right commutes.

$$\begin{tikzcd}
\mathsf{Var}_\Sigma \arrow[r, "\mathbf{T}_\Sigma"] \arrow[d, "\mathbf{L}_\varphi"'] & \mathsf{Set}^S_d \arrow[d, "\mathbf{S}^d_\varphi"] \arrow[dl, Rightarrow, "\overline{\varphi}"'] \\
\mathsf{Var}_{\Sigma'} \arrow[r, "\mathbf{T}_{\Sigma'}"'] & \mathsf{Set}^{S'}_d
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{S}^d_\varphi(T_\Sigma(X)) \arrow[r, "\mathbf{S}^d_\varphi(\boldsymbol{t}^*)"] \arrow[d, "\overline{\varphi}_X"'] & \mathbf{S}^d_\varphi(T_\Sigma(Y)) \arrow[d, "\overline{\varphi}_Y"] \\
T_{\Sigma'}(\mathbf{S}^d_\varphi(X)) \arrow[r, "\mathbf{L}_\varphi(\boldsymbol{t})^*"] & T_{\Sigma'}(\mathbf{S}^d_\varphi(Y))
\end{tikzcd}$$

Here we have another situation, as mentioned in Remark 19, where term algebra adjunctions are of no help. Since the S'-maps $\overline{\varphi}_X : \mathbf{S}^d_\varphi(T_\Sigma(X)) \to T_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ are defined by induction over $\mathbf{S}^d_\varphi(T_\Sigma(X))$, also the commutativity of the square has to be and can be shown by induction over $\mathbf{S}^d_\varphi(T_\Sigma(X))$ (compare Remark 11).

Since the functors $\mathbf{L}_\varphi : \mathsf{Var}_\Sigma \to \mathsf{Var}_{\Sigma'}$ and $\mathbf{T}_\Sigma : \mathsf{Var}_\Sigma \to \mathsf{Set}^S_d$ are simply defined be extending the maps between objects $\mathbf{S}^d_\varphi : Var_S \to Var_{S'}$ and $T_\Sigma : Var_S \to \mathsf{Set}^S_d$ to morphisms, the validity of Lemma 1 is preserved.

Corollary 2 (Composition of Term Translations)**.** *For any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ and the corresponding translations of terms $\overline{\varphi} : \mathbf{T}_\Sigma \,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{L}_\varphi \,;\mathbf{T}_{\Sigma'} : \mathsf{Var}_\Sigma \to \mathsf{Set}^{S'}_d$, $\overline{\psi} : \mathbf{T}_{\Sigma'} \,;\mathbf{S}^d_\psi \Rightarrow \mathbf{L}_\psi \,;\mathbf{T}_{\Sigma''} : \mathsf{Var}_{\Sigma'} \to \mathsf{Set}^{S''}_d$ we do have:*

$$\overline{\varphi;\psi} = (\overline{\varphi}.\,\mathbf{S}^d_\psi);(\mathbf{L}_\varphi.\,\overline{\psi}) : \mathbf{T}_\Sigma \,;\mathbf{S}^d_{\varphi;\psi} \Rightarrow \mathbf{L}_{\varphi;\psi} \,;\mathbf{T}_{\Sigma''} : \mathsf{Var}_\Sigma \to \mathsf{Set}^{S''}_d .$$

$$\begin{tikzcd}
\mathsf{Var}_\Sigma \arrow[r, "\mathbf{T}_\Sigma"] \arrow[d, "\mathbf{L}_\varphi"'] & \mathsf{Set}^S_d \arrow[d, "\mathbf{S}^d_\varphi"] \arrow[dl, Rightarrow, "\overline{\varphi}"'] \\
\mathsf{Var}_{\Sigma'} \arrow[r, "\mathbf{T}_{\Sigma'}"] \arrow[d, "\mathbf{L}_\psi"'] & \mathsf{Set}^{S'}_d \arrow[d, "\mathbf{S}^d_\psi"] \arrow[dl, Rightarrow, "\overline{\psi}"'] \\
\mathsf{Var}_{\Sigma''} \arrow[r, "\mathbf{T}_{\Sigma''}"] & \mathsf{Set}^{S''}_d
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Var}_\Sigma \arrow[r, "\mathbf{T}_\Sigma"] \arrow[d, "\mathbf{L}_{\varphi;\psi}"'] & \mathsf{Set}^S_d \arrow[d, "\mathbf{S}^d_{\varphi;\psi}"] \arrow[dl, Rightarrow, "\overline{\varphi;\psi}"'] \\
\mathsf{Var}_{\Sigma''} \arrow[r, "\mathbf{T}_{\Sigma''}"] & \mathsf{Set}^{S''}_d
\end{tikzcd}$$

6.3. Translation of Expressions and Substitution Application

Analogously to terms, we would also like to extend the map $FE_\Sigma : Var_S \to \mathsf{Set}$ to a functor $\mathbf{FE}_\Sigma : \mathsf{Var}_\Sigma \to \mathsf{Set}$. For all Σ-substitutions $\boldsymbol{r} : X \to T_\Sigma(Z)$ we can indeed define a map $\overline{\boldsymbol{r}} : FE_\Sigma(X) \to FE_\Sigma(Z)$ describing the application of the substitution to Σ-expressions on X. Due to the presence of quantification, the assignments $\boldsymbol{r} \mapsto \overline{\boldsymbol{r}}$ will be, however, not functorial "on the nose" as we will discuss below.

There seems to be no way out! As long as we do not utilize something like the *de Bruijn index* [30] to avoid named bound variables, we have to deal, in one or another way, with the problem that substitution application may cause an unintended interaction between free and bound variables. One approach to avoid such an unintended interaction is to inductively define a binary meta-level predicate stating when a substitution is "legal" for a certain expression as it is done, for example, in [4]. From the perspective of deduction, this is viable. We will, however, not adapt this approach, since we would end up with partial substitution maps .

Another approach is to use *α-conversion*, i.e., renaming of bound variables, in such a way, that we obtain total substitution maps. We will define the application of a substitution to expressions, according to the inductive definition of expressions in Definition 5, thereby establishing a stepwise inductive variant of α-conversion.

To meet the "proper inclusion condition" for Quantification in Definition 5, we must use non-symmetric sums when defining substitution application in case of Quantification. That is, for any two S-sets $X_1, X_2 \in Var_S$ of variables we must choose a sum $X_1 \vec{+}_S X_2$ in Set^S_d such that the left injection $\kappa_1 : X_1 \to X_1 \vec{+}_S X_2$ is an S-inclusion $X_1 \subseteq X_1 \vec{+}_S X_2$. If X_1 and X_2 are disjoint, we simply choose $X_1 \vec{+}_S X_2 := X_1 \cup X_2$.

Definition 13 (Substitution Application for Expressions)**.** *For any signature Σ the family of maps $\overline{\boldsymbol{r}} : FE_\Sigma(X) \to FE_\Sigma(Z)$ with $\boldsymbol{r} : X \to T_\Sigma(Z)$ a Σ-substitution is defined inductively and in parallel by the following assignments:*

1. *Atomic expressions:*
   (a) *Equation: $X \rhd (t_1 = t_2) \;\mapsto\; Z \rhd (\boldsymbol{r}^*_s(t_1) = \boldsymbol{r}^*_s(t_2))$ for any $s \in S$ and any $t_1, t_2 \in T_\Sigma(X)_s$ with $\boldsymbol{r}^* : T_\Sigma(X) \to T_\Sigma(Z)$ given by (26).*
   (b) *Relational Atom: $X \rhd \mathrm{p}\,\mathbf{syn}(\boldsymbol{t}) \;\mapsto\; Z \rhd \mathrm{p}\,\mathbf{syn}(\boldsymbol{t};\boldsymbol{r}^*)$ for any $\mathrm{p} \in P$ and any S-map $\boldsymbol{t} : ar(\mathrm{p}) \to T_\Sigma(X)$ with $\boldsymbol{r}^* : T_\Sigma(X) \to T_\Sigma(Z)$ given by (26).*
2. *Everything: $X \rhd \top\langle\rangle \;\mapsto\; Z \rhd \top\langle\rangle$*
3. *Void: $X \rhd \bot\langle\rangle \;\mapsto\; Z \rhd \bot\langle\rangle$*
4. *Conjunction: $X \rhd (Ex_1 \wedge Ex_2) \;\mapsto\; Z \rhd (\overline{\boldsymbol{r}}(Ex_1) \wedge \overline{\boldsymbol{r}}(Ex_2))$*
5. *Disjunction: $X \rhd (Ex_1 \vee Ex_2) \;\mapsto\; Z \rhd (\overline{\boldsymbol{r}}(Ex_1) \vee \overline{\boldsymbol{r}}(Ex_2))$*

6. *Implication:* $X \rhd (Ex_1 \to Ex_2) \;\mapsto\; Z \rhd (\overline{\boldsymbol{r}}(Ex_1) \to \overline{\boldsymbol{r}}(Ex_2))$
7. *Negation:* $X \rhd \neg Ex \;\mapsto\; Z \rhd \neg\, \overline{\boldsymbol{r}}(Ex)$
8. *Quantification:* $X \rhd \exists(\mathbf{syn}(Y\backslash X) : Ex) \;\mapsto\; Z \rhd \exists(\mathbf{syn}(W\backslash Z) : \overline{[\boldsymbol{r};\subset,\kappa_2;\eta_W]}(Ex))$ *and* $X \rhd \forall(\mathbf{syn}(Y\backslash X) : Ex) \;\mapsto\; Z \rhd \forall(\mathbf{syn}(W\backslash Z) : \overline{[\boldsymbol{r};\subset,\kappa_2;\eta_W]}(Ex))$ *for any expression* $Y \rhd Ex$ *and any proper S-inclusion* $X \subset Y$ *where* $W := Z \vec{+}_S (Y \setminus X)$ *and thus* $Z \subset W$*. The extended substitution* $[\boldsymbol{r};\subset,\kappa_2;\eta_W] : Y \to T_\Sigma(W)$ *is constructed as a cotuple. Keep in mind Remark 3 and that* Y *is the sum of* X *and* $Y \setminus X$ *in* Set^S_d *since* $X \subset Y$*.*

$$\begin{tikzcd}
X \arrow[r, "\subset"] \arrow[d, "\boldsymbol{r}"'] & Y \arrow[d, "{[\boldsymbol{r};\subset,\kappa_2;\eta_W]}"] & Y \setminus X \arrow[l, "\supset"'] \arrow[d, "\kappa_2"] \\
T_\Sigma(Z) \arrow[r, "\subset"] & T_\Sigma(W) & W \arrow[l, "\eta_W"']
\end{tikzcd}$$

It looks like there is no choice of non-symmetric sums in Var_S such that the assignments $X \mapsto FE_\Sigma(X)$ and $\boldsymbol{r} \mapsto \overline{\boldsymbol{r}}$ define a functor from Var_Σ into Set since for any Σ-substitutions $\boldsymbol{r}_1 : X_1 \to T_\Sigma(X_2)$, $\boldsymbol{r}_2 : X_2 \to T_\Sigma(X_3)$ the maps $\overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2} : FE_\Sigma(X_1) \to FE_\Sigma(X_3)$ and $\overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*} : FE_\Sigma(X_1) \to FE_\Sigma(X_3)$ coincide, in general, only for Σ-expressions without quantifications. For a Σ-expression Ex on X_1 containing quantifications, the two Σ-expressions $\overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2}(Ex)$ and $\overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*}(Ex)$ may be different. They will be, however, the "same up to α-conversion". Especially, it can be shown that they are semantically equivalent.

Lemma 4 (Semantic Equivalence)**.** *For* Σ*-substitutions* $\boldsymbol{r}_1 : X_1 \to T_\Sigma(X_2)$*,* $\boldsymbol{r}_2 : X_2 \to T_\Sigma(X_3)$ *the maps* $\overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2} : FE_\Sigma(X_1) \to FE_\Sigma(X_3)$ *and* $\overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*} : FE_\Sigma(X_1) \to FE_\Sigma(X_3)$ *are semantically equivalent in the sense that*

$$[\![\, \overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2}(Ex) \,]\!]^{\mathcal{A}}_{X_3} = [\![\, \overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*}(Ex) \,]\!]^{\mathcal{A}}_{X_3} \quad \text{for all } Ex \in FE_\Sigma(X_1) \text{ and all } \Sigma\text{-structures } \mathcal{A}.$$

Remark 20 (α-conversion)**.** *The best we can hope for, if we intend to formalize the observation that the* Σ*-expressions* $\overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2}(Ex)$ *and* $\overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*}(Ex)$ *are the "same up to* α*-conversion", is some kind of "pseudo functor" (maybe in the sense of [31]?). For such a formalization, we should probably upgrade the sets* $FE_\Sigma(X)$ *to small categories* $\mathsf{FE}_\Sigma(X)$ *with morphisms encoding the information that two* Σ*-expressions are the "same up to* α*-conversion". The assignments* $X \mapsto \mathsf{FE}_\Sigma(X)$ *and* $\boldsymbol{r} \mapsto \overline{\boldsymbol{r}}$ *hopefully define then a "pseudo functor" from* Var_Σ *into* Cat *in the sense that for any* Σ*-substitutions* $\boldsymbol{r}_1 : X_1 \to T_\Sigma(X_2)$*,* $\boldsymbol{r}_2 : X_2 \to T_\Sigma(X_3)$ *there is a natural isomorphism between the functors* $\overline{\boldsymbol{r}_1};\overline{\boldsymbol{r}_2} : \mathsf{FE}_\Sigma(X_1) \to \mathsf{FE}_\Sigma(X_3)$ *and* $\overline{\boldsymbol{r}_1;\boldsymbol{r}_2^*} : \mathsf{FE}_\Sigma(X_1) \to \mathsf{FE}_\Sigma(X_3)$*.*

In light of deduction, there is no need for a formalization of α-conversion! It should be quite enough to formally grasp what we have at hand besides Lemma 4.

Due to (27), we obviously have $\overline{\eta_X} = id_{FE_\Sigma(X)} : FE_\Sigma(X) \to FE_\Sigma(X)$ for the canonical substitutions η_X: $X \to T_\Sigma(X)$, thus the assignments $X \mapsto FE_\Sigma(X)$ and $\boldsymbol{r} \mapsto \overline{\boldsymbol{r}}$ define a reflexive graph homomorphism $\mathbf{FE}_\Sigma : gr(\mathsf{Var}_\Sigma) \to gr(\mathsf{Set})$ (see Section 2).

We consider an arbitrary signature morphism $\varphi : \Sigma \to \Sigma'$. To show that the family of maps $\overline{\varphi}_X : FE_\Sigma(X) \to FE_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ with X in $|\mathsf{Var}_\Sigma| = Var_S$, defined in Section 4.2.3, also establishes a natural transformation $\overline{\varphi} : \mathbf{FE}_\Sigma \Rightarrow \mathbf{L}_\varphi \,;\mathbf{FE}_{\Sigma'} : gr(\mathsf{Var}_\Sigma) \to gr(\mathsf{Set})$ between reflexive graph homomorphisms, we have to prove that for any Σ-substitution $\boldsymbol{r} : X \to T_\Sigma(Z)$ the square of maps on the right below commutes.

$$\begin{tikzcd}
gr(\mathsf{Var}_\Sigma) \arrow[r, "\mathbf{FE}_\Sigma"] \arrow[d, "\mathbf{L}_\varphi"'] & gr(\mathsf{Set}) \arrow[dl, phantom, "\overline{\varphi}\Downarrow"] \\
gr(\mathsf{Var}_{\Sigma'}) \arrow[ru, "\mathbf{FE}_{\Sigma'}"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
FE_\Sigma(X) \arrow[r, "\overline{\boldsymbol{r}}"] \arrow[d, "\overline{\varphi}_X"'] & FE_\Sigma(Z) \arrow[d, "\overline{\varphi}_Z"] \\
FE_{\Sigma'}(\mathbf{S}^d_\varphi(X)) \arrow[r, "\overline{\mathbf{L}_\varphi(\boldsymbol{r})}"] & FE_{\Sigma'}(\mathbf{S}^d_\varphi(Z))
\end{tikzcd}$$

The commutativity of the right square can be inductively shown according to the inductive definition of expressions in Definition 5 and the corresponding inductive definition of the maps

$\overline{\varphi}_X : FE_\Sigma(X) \to FE_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ in Section 4.2.3 and of the translation maps $\overline{\boldsymbol{r}} : FE_\Sigma(X) \to FE_\Sigma(Z)$ in Definition 13. The basic case 1.(a) Equation follows from the fact in Section 6.2 that the family of S'-maps $\overline{\varphi}_X : \mathbf{S}^d_\varphi(T_\Sigma(X)) \to T_{\Sigma'}(\mathbf{S}^d_\varphi(X))$ constitutes a natural transformation $\overline{\varphi} : \mathbf{T}_\Sigma\,;\mathbf{S}^d_\varphi \Rightarrow \mathbf{L}_\varphi\,;\mathbf{T}_{\Sigma'} : \mathsf{Var}_\Sigma \to \mathsf{Set}^{S'}_d$. This fact also ensures the second basic case 1.(b) Relational Atom due to the definition of $\mathbf{L}_\varphi : \mathsf{Var}_\Sigma \to \mathsf{Var}_{\Sigma'}$ and the functoriality of $\mathbf{S}^d_\varphi$. The cases 2. Everything to 7. Negation can be straightforwardly proven.

The critical case 8. Quantification can only be shown under the *additional assumption* that for all signature morphisms $\varphi : \Sigma \to \Sigma'$ and all disjoint S-sets X, Y of variables the equation $\mathbf{S}^d_\varphi(X \vec{+}_S Y) = \mathbf{S}^d_\varphi(X) \vec{+}_{S'} \mathbf{S}^d_\varphi(Y)$ holds. Such a coordination of choices of non-symmetric sums can be achieved if we define the choices for all sets Var_S in a uniform way by means of one and the same choice of non-symmetric sums in Set for the set $\wp_{fin}(N_{var})$ of all finite subsets of the set N_{var} of names for variables: For any disjoint S-sets X, Y in Var_S, we consider the corresponding S-typed sets $(\bigcup X, t_X)$, $(\bigcup Y, t_Y)$. We take the chosen non-symmetric sum $\bigcup X \vec{+} \bigcup Y$ of the sets $\bigcup X$ and $\bigcup Y$ in $\wp_{fin}(N_{var})$ and construct its typing morphism as the cotuple $[t_X, t_Y] : \bigcup X \vec{+} \bigcup Y \to S$. The S-typed set $(\bigcup X \vec{+} \bigcup Y, [t_X, t_Y])$ is a non-symmetric sum of $(\bigcup X, t_X)$ and $(\bigcup Y, t_Y)$ in Set/S. The desired non-symmetric sum of X and Y in Var_S can be obtained via the isomorphism $\mathbf{I}_S : \mathsf{Set}/S \to \mathsf{Set}^S_d$:

$$X \vec{+}_S Y := \mathbf{I}_S(\bigcup X \vec{+} \bigcup Y, [t_X, t_Y]) = \{[t_X, t_Y]^{-1}(s) \mid s \in S\}.$$

Case 8. Quantification is then entailed by the definition of $\mathbf{L}_\varphi$ and its functoriality as well as the fact that $\mathbf{S}^d_\varphi$, as a left-adjoint functor, preserves sums.

Analogously to Lemma 1, the validity of Lemma 2 is preserved since the functors $\mathbf{L}_\varphi : \mathsf{Var}_\Sigma \to \mathsf{Var}_{\Sigma'}$ and the reflexive graph homomorphisms $\mathbf{FE}_\Sigma : gr(\mathsf{Var}_\Sigma) \to gr(\mathsf{Set})$ are defined as extensions of maps between between objects to morphisms.

Corollary 3 (Composition of Expression Translations)**.** *For any signature morphisms $\varphi : \Sigma \to \Sigma'$, $\psi : \Sigma' \to \Sigma''$ and the corresponding translations of expressions $\overline{\varphi} : \mathbf{FE}_\Sigma \Rightarrow \mathbf{L}_\varphi\,;\mathbf{FE}_{\Sigma'} : gr(\mathsf{Var}_\Sigma) \to gr(\mathsf{Set})$ and $\overline{\psi} : \mathbf{FE}_{\Sigma'} \Rightarrow \mathbf{L}_\psi\,;\mathbf{FE}_{\Sigma''} : gr(\mathsf{Var}_{\Sigma'}) \to gr(\mathsf{Set})$ we do have:*

$$\overline{\varphi;\psi} = \overline{\varphi}; (\mathbf{L}_\varphi.\,\overline{\psi}) : \mathbf{FE}_\Sigma \Rightarrow \mathbf{L}_{\varphi;\psi}\,;\mathbf{FE}_{\Sigma''} : gr(\mathsf{Var}_\Sigma) \to gr(\mathsf{Set}).$$

$$\begin{tikzcd}
gr(\mathsf{Var}_\Sigma) \arrow[d, "\mathbf{L}_\varphi"'] \arrow[drr, "\mathbf{FE}_\Sigma"] & {} \arrow[d, Rightarrow, "\overline{\varphi}"'] & \\
gr(\mathsf{Var}_{\Sigma'}) \arrow[rr, "\mathbf{FE}_{\Sigma'}"'] \arrow[d, "\mathbf{L}_\psi"'] & {} \arrow[d, Rightarrow, "\overline{\psi}"'] & gr(\mathsf{Set}) \\
gr(\mathsf{Var}_{\Sigma''}) \arrow[urr, "\mathbf{FE}_{\Sigma''}"'] & {} &
\end{tikzcd}
\qquad
\begin{tikzcd}
gr(\mathsf{Var}_\Sigma) \arrow[dd, "\mathbf{L}_{\varphi;\psi}"'] \arrow[drr, "\mathbf{FE}_\Sigma"] & {} \arrow[dd, Rightarrow, "\overline{\varphi;\psi}"'] & \\
 & & gr(\mathsf{Set}) \\
gr(\mathsf{Var}_{\Sigma''}) \arrow[urr, "\mathbf{FE}_{\Sigma''}"'] & {} &
\end{tikzcd}$$

6.4. Translation of Expressions vs Derived Operations

According to (4), any Σ-substitution $\boldsymbol{t} : X \to T_\Sigma(Y)$ represents for any Σ-structure $\mathcal{A}$ a derived operation $\boldsymbol{t}^{\mathcal{A}} : A^Y \to A^X$ in the opposite direction with $\boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}) := \boldsymbol{t}; \boldsymbol{a}^\circ$ for all assignments $\boldsymbol{a} : Y \to A$. For any Σ-substitution $\boldsymbol{r} : Y \to T_\Sigma(Z)$ and any assignment $\boldsymbol{b} : Z \to A$ it can also inductively shown that

$$(\boldsymbol{r}; \boldsymbol{b}^\circ)^\circ = \boldsymbol{r}^*; \boldsymbol{b}^\circ \;: T_\Sigma(Y) \to A\,. \tag{28}$$

This entails

$$(\boldsymbol{t};^\Sigma \boldsymbol{r})^{\mathcal{A}} = (\boldsymbol{t}; \boldsymbol{r}^*)^{\mathcal{A}} = \boldsymbol{r}^{\mathcal{A}}; \boldsymbol{t}^{\mathcal{A}} \;: A^Z \to A^X\,. \tag{29}$$

That is, composition in Var_Σ syntactically represents the composition of derived operations! (29) and (5) ensure that for any Σ-structure $\mathcal{A}$ the assignments $X \mapsto A^X$ and $\boldsymbol{t} \mapsto \boldsymbol{t}^{\mathcal{A}}$ define a contra-

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 February 2025 doi:10.20944/preprints202502.0785.v1

variant functor $\mathbf{Op}^{\mathcal{A}}_{\Sigma} : \mathsf{Var}^{op}_{\Sigma} \to \mathsf{Set}$ the semantic counterpart of the reflexive graph homomorphisms $\mathbf{FE}_{\Sigma} : gr(\mathsf{Var}_{\Sigma}) \to gr(\mathsf{Set})$.

$\mathbf{Op}^{\mathcal{A}}_{\Sigma}$ and $\mathbf{FE}_{\Sigma}$ are not really matching the "original institution pattern". Nevertheless, we do have a corresponding satisfaction condition: For any Σ-substitution $\boldsymbol{t} : X \to T_{\Sigma}(Y)$, any Σ-expression $X \triangleright Ex$, any Σ-structure $\mathcal{A}$ and any assignment $\boldsymbol{a} \in A^{Y}$ it holds that

$$\boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}) \models^{\Sigma}_{\mathcal{A}} X \triangleright Ex \, (\text{i.e. } \boldsymbol{t}^{\mathcal{A}}(\boldsymbol{a}) \in [\![\, Ex \,]\!]^{\mathcal{A}}_{X}) \quad \text{iff} \quad \boldsymbol{a} \models^{\Sigma}_{\mathcal{A}} Y \triangleright \overline{\boldsymbol{t}}(Ex) \, (\text{i.e. } \boldsymbol{a} \in [\![\, \overline{\boldsymbol{t}}(Ex) \,]\!]^{\mathcal{A}}_{Y}). \tag{30}$$

This is our reconstruction of the statement in Proposition 5.6 in [17].

6.5. Statements in Context and Substitutions

As we have seen, Σ-substitutions $\boldsymbol{t} : X \to T_{\Sigma}(Y)$ establish directed relationships between Σ-expressions $X \triangleright Ex$ and $Y \triangleright \overline{\boldsymbol{t}}(Ex)$. An obvious idea could be to extend now the sets $Stm_{\Sigma}(K)$ of all Σ-statements in contexts K in Definition 10 by morphisms of the form $\boldsymbol{t} : (X, Ex, ??) \to (Y, \overline{\boldsymbol{t}}(Ex), \gamma)$ with $\gamma \colon X \to K$. What should "??" be in this case? A natural proposal would be to choose the S-map $\boldsymbol{t}; (\gamma; \eta_K)^* : X \to T_{\Sigma}(K)$.

$$\begin{tikzcd}
X \arrow[r, "\boldsymbol{t}"] & T_{\Sigma}(Y) \arrow[d, "(\gamma;\eta_K)^*"'] & Y \arrow[l, "\eta_Y"'] \arrow[d, "\gamma"] \\
& T_{\Sigma}(K) & K \arrow[l, "\eta_K"']
\end{tikzcd}$$

That is, if we want to introduce morphisms between statements induced by substitutions, we probably must upgrade the concept of "binding" to S-maps of the form $X \to T_{\Sigma}(K)$. In such a way, we would get morphisms like $\boldsymbol{t} : (X, Ex, \boldsymbol{t}; \rho^*) \to (Y, \overline{\boldsymbol{t}}(Ex), \rho)$ with $\boldsymbol{t} : X \to T_{\Sigma}(Y)$, $\rho : Y \to T_{\Sigma}(K)$ and thus $\boldsymbol{t}; \rho^* : X \to T_{\Sigma}(K)$.

As seen in Section 6.3, the composition of substitutions will, however, not establish a category since composition will only be associative "up to α-conversions". At the moment, we do not have the capacity and/or the technical skills to find out if we would get something like a "pseudo category", in the sense of [31], a "quasi category" [32] or something else.

On the other side, we already said, that a formalization of α-conversion does not seem to be relevant for deduction, thus we could be more pragmatic and lift up the set $Stm_{\Sigma}(K)$ to a reflexive graph $\mathsf{Stm}_{\Sigma}(K)$ with substitutions as arrows and with a composition operation satisfying the identity law but satisfying the associativity law only "up to semantic equivalence".

At present stage we can not overlook if such a promotion of $Stm_{\Sigma}(K)$ to $\mathsf{Stm}_{\Sigma}(K)$ is feasible and/or necessary. First, we should study and investigate, in detail, first-order deduction in view of Logics of Statements in Context, in a prospective follower paper, to find this out! It may be also beneficial to add morphisms of the form $K \to T_{\Sigma}(G)$ to the category Cxt_S of contexts, once we decide to utilize bindings of the form $X \to T_{\Sigma}(K)$?

For now, we only observe that the satisfaction condition in (30) ensures that the statements $(X, Ex, \boldsymbol{t}; \rho^*)$ and $(Y, \overline{\boldsymbol{t}}(Ex), \rho)$ are semantically equivalent. For any Σ-interpretation $(\iota, \mathcal{A})$ of a context K in a Σ-structure $\mathcal{A}$ we have $(\iota, \mathcal{A}) \models^{\Sigma}_{K} (X, Ex, \boldsymbol{t}; \rho^*)$ if, and only if, $(\iota, \mathcal{A}) \models^{\Sigma}_{K} (Y, \overline{\boldsymbol{t}}(Ex), \rho)$ due to Definition 12, (28) and (27).

This semantic equivalence also indicates that there is no need for bindings of the form $\varrho : X \to T_{\Sigma}(K)$ if we are only concerned about specifications and not about deduction. For any such binding there is an S-set Y, a Σ-substitution $\boldsymbol{t} : X \to T_{\Sigma}(Y)$ and a simple injective binding $\gamma \colon Y \to K$ such that $\varrho = \boldsymbol{t}; (\gamma; \eta_K)^*$. That is, for any Σ-statement (X, Ex, ϱ) with a more advanced binding $\varrho : X \to T_{\Sigma}(K)$ there is a semantically equivalent Σ-statement $(Y, \overline{\boldsymbol{t}}(Ex), \gamma)$ with a simple binding $\gamma \colon Y \to K$.

To construct such a factorization of $\varrho : X \to T_{\Sigma}(K)$ we consider the minimal S-subset $K' \subseteq K$ such that $\varrho(X) \subseteq T_{\Sigma}(K') \subseteq T_{\Sigma}(K)$ (see Remark 3). In such a way, we factorize $\varrho : X \to T_{\Sigma}(K)$ into an S-map $\varrho' : X \to T_{\Sigma}(K')$ and the inclusion $T_{\Sigma}(K') \subseteq T_{\Sigma}(K)$. Since K' is finite, we can choose an S-set Y

of variables with a bijective S-map $i_Y : Y \to K'$ and thus a bijective S-map $j_Y : T_\Sigma(K') \to T_\Sigma(Y)$. We set $\boldsymbol{t} := \varrho'; j_Y$ and obtain $\gamma : Y \to K$ as the composition of $i_Y : Y \to K'$ and the inclusion $K' \subseteq K$.

7. Sketches and Diagrams

7.1. Sketches and Sketch Implications

We start with an informal discussion. Definition 12 declares the validity of a single Σ-statement in context K for single interpretations of context K in any Σ-structure $\mathcal{A}$. This allows us, in analogy to Σ-expressions, to define for a Σ-statement its semantics in a certain Σ-structure $\mathcal{A}$ as the set of all valid interpretations of K in $\mathcal{A}$. This does not give us, however, a concept of validity of the Σ-statement in $\mathcal{A}$ at hand!

To come up with a reasonable concept of validity of a Σ-statement in a Σ-structure, let us have a look back at the paradigmatic difference between an equation $t = r$ and a conditional equation $(X : \longrightarrow t = r)$ with an empty premise [7,33]. Our main methodological point is to consider a conditional equation $(X : t_1 = r_1, \ldots, t_n = r_n \longrightarrow t = r)$ not as a lazy notation for the closed expression $\forall(X : t_1 = r_1 \wedge \ldots \wedge t_n = r_n \to t = r)$ but as an "arrow" between sets of equations established by an identity context morphism: $id_X : (X : \{t_1 = r_1, \ldots, t_n = r_n\}) \longrightarrow (X : \{t = r\})$. A conditional equation is valid in a Σ-structure $\mathcal{A}$ if, and only if, each solution of the premise $(X : \{t_1 = r_1, \ldots, t_n = r_n\})$ is also a solution of the conclusion $(X : \{t = r\})$. This viewpoint obviously scales up to infinite sets X and infinite sets of equations! Since any interpretation of X in $\mathcal{A}$ is a solution of the empty premise, the validity of $id_X : (X : \emptyset) \longrightarrow (X : \{t = r\})$ in $\mathcal{A}$ means nothing but that any interpretation of X in $\mathcal{A}$ is a solution of $t = r$ in $\mathcal{A}$.

As already mentioned at the end of Section 4 and discussed in Remark 16, it is well-known that "empty sorts" potentially cause the problem that deduction rules for one-sorted first-order logic are, in general, not sound for many-sorted deduction [17,19]. The proposal in [19], to solve this problem for many-sorted equational logic, is subsumed in our proposal of Logics of Statements in Contexts: Explicit declaration of variables with their sorts and considering equations as corresponding conditional equations with empty premises.

A similar situation we meet in traditional first-order logic, where the "provable formulas" are exactly the "admissible rules" with an empty premise [4].

In [1] we borrowed the term "sketch" from [9,10,34] to denote a context together with an arbitrary set of statements in this context. This and other definitions can easily be adapted and extended to our First-Order Logics of Statements in Context.

For any signature $\Sigma = (S, P, F, ar, in, out)$ a **Σ- sketch** $\mathbb{K} = (K, St^{\mathbb{K}})$ is given by a context $K \in |\mathsf{Cxt}_S| = |\mathsf{Set}^S_d|$ and a set $St^{\mathbb{K}} \subseteq Stm_\Sigma(K)$ of Σ-statements in context K. $\mathbb{K} = (K, St^{\mathbb{K}})$ is called **atomic** if all the Σ-statements (X, Ex, γ) in $St^{\mathbb{K}}$ are atomic meaning that Ex is an atomic expression, i.e., an equation or a relational atom according to Def. 5.

An interpretation $\iota : K \to A$ of context K in a Σ-structure $\mathcal{A}$ is **a valid interpretation** of the sketch $\mathbb{K} = (K, St^{\mathbb{K}})$ in $\mathcal{A}$, $(\iota, \mathcal{A}) \models^\Sigma_K St^{\mathbb{K}}$ in symbols, if, and only if, $(\iota, \mathcal{A}) \models^\Sigma_K (X, Ex, \gamma)$, i.e., $\gamma; \iota \in [\![Ex]\!]^{\mathcal{A}}_X$ due to (17), for all Σ-statements (X, Ex, γ) in $St^{\mathbb{K}}$. Be aware that the Σ-statements in $St^{\mathbb{K}}$ may have different S-sets X of variables!

A **Σ-sketch implication** $\mathbb{P} \overset{\mu}{\Rightarrow} \mathbb{C}$ is given by two Σ-sketches $\mathbb{P} = (P, St^{\mathbb{P}})$, $\mathbb{C} = (C, St^{\mathbb{C}})$ and a context morphism $\mu : P \to C$. $\mathsf{Sk}(\Sigma)$ denotes the category of all Σ-sketches and all Σ-sketch implications. If $P = C$ and $\mu = id_P$, we simply write $\mathbb{P} \Rightarrow \mathbb{C}$ instead of $\mathbb{P} \overset{id_P}{\Longrightarrow} \mathbb{C}$.

A Σ-sketch implication $\mathbb{P} \overset{\mu}{\Rightarrow} \mathbb{C}$ is **valid in a Σ-structure** $\mathcal{A}$, $\mathcal{A} \models^{\Sigma} \mathbb{P} \overset{\mu}{\Rightarrow} \mathbb{C}$ in symbols, if, and only if, for all interpretations $(\iota, \mathcal{A})$ of P in $\mathcal{A}$ it holds that $(\iota, \mathcal{A}) \models^{\Sigma}_{P} St^{\mathbb{P}}$ implies the existence of an interpretation $(\vartheta, \mathcal{A})$ of context C in $\mathcal{A}$ with $\mu; \vartheta = \iota$ such that $(\vartheta,\ \mathcal{A}) \models^{\Sigma}_{C} St^{\mathbb{C}}$.

$$\begin{tikzcd}
P \arrow[rr, "\mu"] \arrow[dr, "(\iota,\mathcal{A})\models^{\Sigma}_{P}St^{\mathbb{P}}"'] & \arrow[d, phantom, "="] & C \arrow[dl, "\exists(\vartheta,\mathcal{A})\models^{\Sigma}_{C}St^{\mathbb{C}}"] \\
 & U &
\end{tikzcd}$$

Obviously, we trivially have $\mathcal{A} \models^{\Sigma} \mathbb{P} \overset{\mu}{\Rightarrow} \mathbb{C}$ if there is no interpretation of context P in $\mathcal{A}$ at all. As discussed before, this may be the case in many-sorted logics due to the potential presence of empty sorts.

In case $P = C$ and $\mu = id_P$, we have $\mathcal{A} \models^{\Sigma} \mathbb{P} \Rightarrow \mathbb{C}$ if, and only if, each valid interpretation of the premise $\mathbb{P}$ is also a valid interpretation of the conclusion $\mathbb{C}$. In analogy to (conditional) equations, we could now define that a Σ-sketch $\mathbb{C}$ is valid in a Σ-structure $\mathcal{A}$ if, and only if, $\mathcal{A} \models^{\Sigma} (C, \emptyset) \Rightarrow \mathbb{C}$, i.e., if all interpretations $(\iota, \mathcal{A})$ of the context C in $\mathcal{A}$ are valid interpretations of the sketch $\mathbb{C} = (C, St^{\mathbb{C}})$ in $\mathcal{A}$.

7.2. Diagrams

We became familiar with diagrams not via Model Theory [2,3,17] but as a technique used in the construction of free algebras [6,7,33]. From this perspective diagrams arise as syntactic representations of structures. To obtain such a representation, we need, first, a mechanism to syntactically represent the elements of a given structure.

A widely used mechanism is to add for a given Σ-structure $\mathcal{A}$ "copies" of all the elements in the carrier A of $\mathcal{A}$ as new "auxiliary" constant symbols to the signature Σ (thus blowing up Σ to an infinite entity in most cases) [2,6,17]. In contrast, Logics of Statements in Context generalize the approach in [7,33] which adapts the idea of "generators" and "defining relations" from Group Theory.

The elements of contexts are for us, in principal, neither constant symbols nor variables nor elements of carrier sets. Or, to put it the other way around: Of course we can use one and the same entity in different roles, but we should always be aware of the different roles and changes of roles! Contexts establish a bridge between syntax (constant symbols, variables) and semantics (elements of carrier sets), and we use them to define diagrams.

We require contexts to be disjoint S-sets in order to be able to translate Σ-sketches along signature morphisms. This is one of the crucial technical mechanisms to describe the construction of free structures! In contrast, carriers of Σ-structures are not required to be disjoint! So, the first step in transforming a Σ-structure $\mathcal{A}$ into a Σ-sketch will be to transform the carrier A of $\mathcal{A}$ into a disjoint S-set. Note, that the potential necessity of such an initial step has not been observed in [1].

For this first step we define a functor $\mathbf{D}_S : \mathsf{Set}^S \to \mathsf{Set}^S_d$ with $\mathbf{D}_S(A) := A$ for all disjoint S-sets A and $\mathbf{D}_S(A) := (\{(a,s) \mid a \in A_s\} \mid s \in S)$ for all non-disjoint S-sets A. The assignments $(a,s) \mapsto a$ define an isomorphism $\varepsilon_A : \mathbf{D}_S(A) \to A$ in Set^S for any non-disjoint S-set A. For disjoint S-sets A we simply set $\varepsilon_A := id_A : A \to A$ thus the assignments $A \mapsto \varepsilon_A$ establish, in turn, a natural isomorphism $\varepsilon : \mathbf{D}_S; \mathbf{E}_S \Rightarrow id_{\mathsf{Set}^S}$ for the inclusion functor $\mathbf{E}_S : \mathsf{Set}^S_d \hookrightarrow \mathsf{Set}^S$. Moreover, we have $\mathbf{E}_S; \mathbf{D}_S = id_{\mathsf{Set}^S_d}$ thus we actually obtain an equivalence between the categories Set^S and Set^S_d.

In the literature we essentially find two variants of diagrams - an atomic variant [3,17] and a full variant [2]. Obviously, we can define two corresponding variants of sketch encodings of a Σ-structure $\mathcal{A}$. The atomic variant is given by the Σ-sketch $\mathbb{S}^{\mathcal{A}}_{at} := (\mathbf{D}_S(A), St^{\mathcal{A}}_{at})$ with context $\mathbf{D}_S(A)$ and $St^{\mathcal{A}}_{at}$ the set of all atomic (!) Σ-statements (X, Ex, γ) such that $(\varepsilon_A, \mathcal{A}) \models^{\Sigma}_{\mathbf{D}_S(A)} (X, Ex, \gamma)$ for the canonical interpretation $(\varepsilon_A, \mathcal{A})$. The full variant $\mathbb{S}^{\mathcal{A}} = (\mathbf{D}_S(A), St^{\mathcal{A}})$ of the Σ-sketch encoding of $\mathcal{A}$ is simply obtained by droping the restriction "atomic". The interest reader may have a look in [1] for more concepts and results around sketch encodings of structures. Especially, we discuss there a "universal property" of the canonical interpretation $(\varepsilon_A, \mathcal{A})$ which is strongly related to Proposition 4.10 in [17].

To make this section round, we draw the reader's attention to *minimal sketch encodings*. The minimal Σ-sketch encoding $\mathbb{S}^{\mathcal{A}}_{min} = (\mathbf{D}_S(A), St^{\mathcal{A}}_{min})$ of a Σ-structure $\mathcal{A}$ is given by a proper subset $St^{\mathcal{A}}_{min} \subset St^{\mathcal{A}}_{at}$ that only and directly encodes the semantics $\mathtt{p}^{\mathcal{A}} \subseteq A^{ar(\mathtt{p})}$ of all predicate symbols $\mathtt{p} \in P$ in $\mathcal{A}$ and the semantics $\mathtt{op}^{\mathcal{A}} : A^{in(\mathtt{op})} \to A^{out(\mathtt{op})}$ of all operation symbols $\mathtt{op} \in F$ in $\mathcal{A}$: For any predicate symbol $\mathtt{p} \in P$ we assume w.l.o.g. that $\bigcup ar(\mathtt{p}) = \{x_1, \ldots, x_m\}$ thus we have $\mathbf{syn}(\eta_{ar(\mathtt{p})}) = \langle x_1, ..., x_m \rangle$ for the S-map $\eta_{ar(\mathtt{p})} : ar(\mathtt{p}) \to T_\Sigma(ar(\mathtt{p}))$ (see Remarks 2, 4 and 8). We add to $St^{\mathcal{A}}_{min}$ all Σ-statements

$$(ar(\mathtt{p}), \mathtt{p}\langle x_1, ..., x_m \rangle, \gamma : ar(\mathtt{p}) \to \mathbf{D}_S(A)) \quad \text{with} \quad \gamma;\varepsilon_A \in \mathtt{p}^{\mathcal{A}}.$$

For any operation symbol $\mathtt{op} \in F$ we assume w.l.o.g. that $\bigcup in(\mathtt{op}) = \{x_1, \ldots, x_n\}$ and $\bigcup out(\mathtt{op}) = \{y\}$ (see Remark 2 and the "reborn remark" after Definition 4). We consider the Σ-equation $in(\mathtt{op}) \cup out(\mathtt{op}) \triangleright (\mathtt{op}\langle x_1, \ldots, x_n \rangle = y)$ with $\mathtt{op}\langle x_1, \ldots, x_n \rangle, y \in T_\Sigma(in(\mathtt{op}) \cup out(\mathtt{op}))$ and add to $St^{\mathcal{A}}_{min}$ all Σ-statements

$$(in(\mathtt{op}) \cup out(\mathtt{op}), (\mathtt{op}\langle x_1, \ldots, x_n \rangle = y), \gamma : in(\mathtt{op}) \cup out(\mathtt{op}) \to \mathbf{D}_S(A))$$

with $\gamma(y) = \varepsilon_A^{-1}(\mathtt{op}^{\mathcal{A}}(\varepsilon_A(\gamma(x_1)), \ldots, \varepsilon_A(\gamma(x_n))))$. $in(\mathtt{op}) \cup out(\mathtt{op})$ is the sum of $in(\mathtt{op})$ and $out(\mathtt{op})$ in Set^S thus we can utilize the cotuple notation and more abstractly formulate: For any S-map $\gamma_{in} : in(\mathtt{op}) \to \mathbf{D}_S(A)$ add to $St^{\mathcal{A}}_{min}$ the Σ-statement

$$(in(\mathtt{op}) \cup out(\mathtt{op}), (\mathtt{op}\langle x_1, \ldots, x_n \rangle = y), [\gamma_{in}\,,\, \mathtt{op}^{\mathcal{A}}(\gamma_{in};\varepsilon_A);\varepsilon_A^{-1}])$$

We experienced that in some very special cases only the minimal encoding enables us to construct free structures.

8. Conclusions and Further Work

We accomplished one of our objectives anticipated in [1]. We constructed a conservative extension of the traditional institution of many-sorted first-order logic to a First-Order Logic of Statements in Context, comprising not only "closed" but also "open formulas".

This new logic was presented in two versions. First, we constructed a comorphism based indexed institution which was then transformed into a fibred institution by means of an appropriate Grothendieck construction.

In the special case of traditional first-order logic, which is based on the category Set, we do have also operations in contrast to the version of Logics of Statements in Context, presented in [1] for arbitrary (!) categories, where we only considered predicates. So, it was the first time that we had to investigate and formalize the interplay of predicates and operations within the framework of Logics of Statements in Context.

Logics of Statements in Context distinguish, in a consistent and clean way, between syntactic entities, like constant symbols, variables and terms, on one side, and semantic entities, like constants, elements of carrier sets and operations, on the other side. One of the roles for the new entity *context* is to establish the necessary bridge between syntax and semantics.

This kind of inherent "separation of concerns" enabled us to develop the new logics in a very systematic and well-structured way including a relatively extensive investigation of substitutions and substitution applications. We revealed some intrinsic structures and relations between them. Especially, we elucidated the crucial role of the requirement that many-sorted sets of variables as well as many-sorted sets of terms should be disjoint.

Besides the incorporation of operations, some other changes, variations and/or adaptions of the first outline of Logics of Statements in Context in [1] turned out to be useful or even necessary:

1. If we incorporate substitutions as morphisms, Var is no longer a subcategory of Base, but an extension of a subcategory of Base.

2. For quantifications we may use only a certain class of morphisms in Var, or we may use morphisms between objects in Var that are not morphisms in Var at all.
3. The same may happen with Cxt in some cases. We may allow substitutions as binding morphisms but do not consider substitutions between contexts. An open question is then in what categories all the different kinds of substitutions live and how the interaction between assignments, evaluations, substitutions and substitution applications is organized. Maybe we need some comma categories to describe this interaction?
4. In [1] we assumed Carr $\sqsubseteq$ Cxt to define diagrams. In first-order logic we met, however, the situation Carr $\sqsupset$ Cxt. We repaired this by constructing a functor $\mathbf{D} : \mathsf{Carr} \to \mathsf{Cxt}$ establishing, together with the inclusion Cxt $\sqsubset$ Carr, an equivalence of the categories Carr and Cxt.
5. We discussed in Section 6.5 that substitutions allow us to upgrade the sets $Stm_\Sigma(K)$ of statements in contexts, according to [1], to structures $\mathsf{Stm}_\Sigma(K)$ which nearly are categories. "Nearly", in the sense, that composition is only associative "up to semantic equivalence" of statements. If we want to take advantage of this observation, we must leave the big and safe building based upon the original definition of institutions, since we will have sentence functors with other codomains than simply Set.

To extend, consolidate and further develop the open framework of Logics of Statements in Context, we envisage two projects as successors of the present paper:

First-Order Deduction As a vital complement, we should finally analyze and describe deduction in traditional first-order logic in view of Logics of Statements in Context. Especially, we are longing to comprehend the relationship between traditional deduction rules, on one side, and contexts and sketch implications, on the other side.

Horn Logics As indicated in Section 7.1, we do not perceive conditional equations (and analogously Horn clauses) as "first-order formulas" but as implications between atomic sketches. The present paper will hopefully enable us to analyze these two special cases of Horn Logics in detail, and to lift up the insights of the analysis to a more abstract account of Horn Logics in general.

Funding: This research was funded by L. Meltzer Høyskolefond 2024 grant number 1201-32425.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Wolter, U. Logics of Statements in Context - Category Independent Basics. *Mathematics* **2022**, *10*. https://doi.org/10.3390/math10071085.
2. Chang, C.C.; Keisler, H.J. *Model Theory*; Studies in Logic and the Foundations of Mathematics, Elsevier Science, 1990.
3. van Dalen, D. *Logic and Structure*, 5 ed.; Universitext, Springer, 2013. https://doi.org/10.1007/978-1-4471-4558-5.
4. Walicki, M. *Introduction to Mathematical Logic*; World Scientific, 2012.
5. Wolfgang Wechler. Universal Algebra for Computer Scientists. In Proceedings of the Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin Heidelberg, 1992, Vol. 25, p. 339. https://doi.org/10.1007/978-3-642-76771-5.
6. Ehrig, H.; Mahr, B. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*; EATCS Monographs on Theoretical Computer Science 6, Springer, 1985.
7. Reichel, H. *Initial Computability, Algebraic Specifications, and Partial Algebras*; Oxford University Press, 1987.
8. Barr, M.; Wells, C. *Category Theory for Computing Science*; Series in Computer Science, Prentice Hall International: London, 1990.
9. Makkai, M. Generalized Sketches as a Framework for Completeness Theorems. *Journal of Pure and Applied Algebra* **1997**, *115*, 49–79, 179–212, 214–274.
10. Cadish, B.; Diskin, Z. Heterogeneous View Integration via Sketches and Equations. In Proceedings of the ISMIS, Springer, Heidelberg, 1996; Vol. 1079, pp. 603–612.

11. 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.
12. Baader, F.; Horrocks, I.; Sattler, U., Handbook of Knowledge Representation; Elsevier, 2007; chapter 3 Description Logics.
13. Ehrig, H.; Ehrig, K.; Prange, U.; Taentzer, G. *Fundamentals of Algebraic Graph Transformation*; Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2006.
14. Wolter, U.; Diskin, Z.; König, H. Graph Operations and Free Graph Algebras. In Proceedings of the Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig. Springer, LNCS 10800, 2018, pp. 313–331. https://doi.org/10.1007/978-3-319-75396-6_17.
15. Wolter, U.; Truong, T.T. Graph Algebras and Derived Graph Operations. *Logics* **2023**, *1*, 182–239. https://doi.org/10.3390/logics1040010.
16. Goguen, J.A.; Burstall, R.M. Institutions: Abstract Model Theory for Specification and Programming. *Journals of the ACM* **1992**, *39*, 95–146.
17. Diaconescu, R. *Institution-Independent Model Theory*, 1st ed.; Birkhäuser Basel, 2008.
18. Diaconescu, R. Grothendieck Institutions. *Applied Categorical Structures* **2002**, *10*, 383–402.
19. Goguen, J.A.; Meseguer, J. Completeness of Many-Sorted Equational Logic. *Houston Journal of Mathematics* **1985**, *11*, 307 – 334.
20. Barendregt, H.P. *The Lambda Calculus: Its Syntax and Semantics*; North Holland, 1984.
21. Lawvere, F.W. Functorial Semantics of Algebraic Theories. In Proceedings of the Proc. National Academy of Science, U.S.A., 50. Columbia University, 1963, pp. 869–872.
22. Lawvere, F.W. Functorial Semantics of Algebraic Theories and Some Algebraic Problems in the context of Functorial Semantics of Algebraic Theories. *Reprints in Theory and Applications of Categories* **2004**, *5*, 1 – 121. Originally published as Ph.D. thesis, Columbia University, 1963 and in Reports of the Midwest Category Seminar II, 1968, 41-61, © Springer-Verlag.
23. Wolter, U.; Klar, M.; Wessäly, R.; Cornelius, F. Four Institutions – A Unified Presentation of Logical Systems for Specification. Technical Report Bericht-Nr. 94-24, 1994.
24. Diskin, Z.; Wolter, U. A Diagrammatic Logic for Object-Oriented Visual Modeling. *ENTCS* **2008**, *203/6*, 19–41. https://doi.org/10.1016/j.entcs.2008.10.041.
25. Martini, A.; Wolter, U.; Haeusler, E.H. Fibred and Indexed Categories for Abstract Model Theory. *Logic Journal of the IGPL* **2007**, *15(5-6)*, 707–739. https://doi.org/doi:10.1093/jigpal/jzm045.
26. Goguen, J.A.; Meseguer, J. Remarks on Remarks on Many-Sorted Equational Logic. *Bulletin of the EATCS* **1986**.
27. Goguen, J.A.; Roşu, G. Institutions Morphisms. *Formal Aspects of Computing* **2002**, *12*, 274–307.
28. Wolter, U.; Martini, A.; Häusler, E. Indexed and fibered structures for partial and total correctness assertions. *Mathematical Structures in Computer Science* **2022**, p. 1–31. https://doi.org/10.1017/S0960129522000275.
29. Poigné, A. Algebra categorically. In *Category Theory and Computer Programming: Tutorial and Workshop, Guildford, UK, 16–20 September 1985 Proceedings*; Pitt, D.; Abramsky, S.; Poigné, A.; Rydeheard, D., Eds.; Springer Berlin Heidelberg: Berlin/Heidelberg, Germany, 1986; pp. 76–102. https://doi.org/10.1007/3-540-17162-2_118.
30. de Bruijn, N.G. Lambda Calculus Notation with Nameless Dummies: A Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem. *Indagationes Mathematicae* **1972**, *34*, 381–392.
31. Ferreira, N.M. Pseudo-categories, 2006, [arXiv:math.CT/math/0604549].
32. Riehl, E. *Categorical Homotopy Theory*; New Mathematical Monographs, Cambridge University Press, 2014. https://doi.org/10.1017/cbo9781107261457.
33. Wolter, U. An Algebraic Approach to Deduction in Equational Partial Horn Theories. *J. Inf. Process. Cybern. EIK* **1990**, *27*, 85–128.
34. 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, 1997, pp. 123–132.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.