mathematics

MDPI

Article

Logics of Statements in Context-Category Independent Basics

Uwe Wolter

Department of Informatics, University of Bergen, 5020 Bergen, Norway; uwe.wolter@uib.no

Abstract: Based on a formalization of open formulas as statements in context, the paper presents a freshly new and abstract view of logics and specification formalisms. Generalizing concepts like sets of generators in Group Theory, underlying graph of a sketch in Category Theory, sets of individual names in Description Logic and underlying graph-based structure of a software model in Software Engineering, we coin an abstract concept of context. We show how to define, in a category independent way, arbitrary first-order statements in arbitrary contexts. Examples of those statements are defining relations in Group Theory, commutative, limit and colimit diagrams in Category Theory, assertional axioms in Description Logic and constraints in Software Engineering. To validate the appropriateness of the newly proposed abstract framework, we prove that our category independent definitions and constructions give us a very broad spectrum of Institutions of Statements at hand. For any Institution of Statements, a specification (presentation) is given by a context together with a set of first-order statements in that context. Since many of our motivating examples are variants of sketches, we will simply use the term sketch for those specifications. We investigate exhaustively different kinds of arrows between sketches and their interrelations. To pave the way for a future development of category independent deduction calculi for sketches, we define arbitrary first-order sketch conditions and corresponding sketch constraints as a generalization of graph conditions and graph constraints, respectively. Sketch constraints are the crucial conceptual tool to describe and reason about the structure of sketches. We close the paper with some vital observations, insights and ideas related to future deduction calculi for sketches. Moreover, we outline that our universal method to define sketch constraints enables us to establish and to work with conceptual hierarchies of sketches.

Keywords: first-order logic; abstract model theory; institution; sketch; algebraic specification; description logic; graph conditions; graph constraints; diagram predicate framework

MSC: 03B70; 03C95; 18C30; 68N30; 68Q65

Citation: Wolter, U. Logics of Statements in Context-Category Independent Basics. *Mathematics* **2022**, *10*, 1085. https://doi.org/10.3390/math10071085

Academic Editor: Răzvan Diaconescu

Received: 1 February 2022
Accepted: 18 March 2022
Published: 28 March 2022

1. Introduction

The impetus towards abstraction is often triggered by the feeling that we do, again and again, the "same thing"—that there are structural similarities between concepts and problems in various areas and on different conceptual levels. We experience facing "similar patterns" when formalizing and reasoning about certain kinds of concepts and problems.

Once we obtain the strong impression that concepts, constructions, proofs and results in various areas and on different conceptual levels are somehow related, we may feel the urge to find out what the commonalities really are and to formalize them in an adequate mathematical language. Naturally, such a formalization will be a pretty abstract one if it should cover a broader range of areas.

In light of these remarks, the paper presents the first stage of expansion of a conceptual framework intended to provide a unified view to a broad range of concepts, constructions and problems we dealt with in our long-standing research in various areas and on different conceptual levels in formal specification. The framework should enable us to describe a

wide range of specification formalisms (modelling techniques) in a uniform way and thus to relate them. Since category theory is the mathematical language of choice to describe and study relations between structures and constructions, we utilize categorical concepts to describe our framework.

1.1. Background, Motivations, Challenges and Principles

In this subsection, we outline different lines of motivation and challenges encouraging us to develop our abstract conceptual framework. In particular, we discuss and try to justify the methodological principles upon which the development of the framework is based.

1.1.1. Universal Algebra and Algebraic Specifications:

We consider a morphism $\varphi : (\Sigma, E) \to (\Sigma', E')$ between two equational specifications, i.e., a signature morphism $\varphi : \Sigma \to \Sigma'$ such that the set E' of Σ'-equations entails the set $\varphi(E)$ of translated Σ-equations. For any (Σ, E)-algebra $\mathcal{A}$ there is a (Σ', E')-algebra $\mathcal{F}_\varphi(\mathcal{A})$ freely generated by $\mathcal{A}$ along φ. The construction of $\mathcal{F}_\varphi(\mathcal{A})$ can be described in four steps: (1) Construct a "syntactic encoding" of $\mathcal{A}$; (2) Translate this syntactic encoding along the signature morphism φ; (3) Use semantic entailment or a deduction calculus to extend the translated encoding of $\mathcal{A}$ to a syntactic encoding of a (Σ', E')-algebra; and (4) Transform the extended encoding into a (Σ', E')-algebra $\mathcal{F}_\varphi(\mathcal{A})$.

There is a widely-used technique to encode a Σ-algebra $\mathcal{A}$ syntactically (see, for example, [1]): The elements of the carrier A of $\mathcal{A}$ are added as auxiliary constants to the signature Σ and the complete behaviour of the operations in $\mathcal{A}$ is encoded by a set $R_\mathcal{A}$ of ground $(\Sigma + A)$-equations. To make the construction of free algebras work, we have to extend (Σ', E') and φ, in such a way that we obtain a signature morphisms $\varphi_\mathcal{A} : (\Sigma + A, E + R_\mathcal{A}) \to (\Sigma' + \varphi(A), E' + \varphi(R_\mathcal{A}))$ and $\mathcal{F}_\varphi(\mathcal{A})$ is constructed as a quotient of the ground $(\Sigma' + \varphi(A))$-term algebra $\mathcal{T}_{\Sigma' + \varphi(A)}(\emptyset)$.

In abstract model theory, this technique is reflected by the idea to define variables by means of signature extensions. (In this paper, we use the term "model" in two conflicting meanings: A "software model", e.g., is a syntactic representation of (certain properties) of a software system (semantic structure) while a "model" in logic is a semantic structure conforming to a formal syntactic description). This is the traditional approach in the theory of institutions (compare [2]). Note that this only works if the signatures in question comprise the concept of operation!

We perceive the above outlined technique in Universal Algebra as not fully adequate. The construction of free algebras becomes unnecessarily involved and we are, unfortunately, forced to work with infinite signatures since there is a kind of circularity in the sense that signatures have to be defined in such a way that the carrier of any potential Σ-algebra for any signature Σ can be encoded by a signature extension of Σ. Somehow, the concept of signature is not a "syntactic" one anymore. In our humble opinion, signatures are (or should be) located on a conceptual level above carriers of algebras. Following the principle of separation of concerns, we would therefore formulate the following first requirement for our framework.

Requirement 1: Define signatures independent of and prior to carriers of algebras.

Adhering to Requirement 1, we will be allowed, for example, to base a specification formalism on finite or enumerable signatures only! Another observation is that signatures are always given by sets, thus we have to adhere to Requirement 1 if we want to work with algebras where the carriers are graphs instead of sets, for example [3].

Generalizing the concept of a group generated by a set of generators and a set of defining relations the small school on Partial Algebraic Specifications in former East-Germany [4–7] developed the concept of a partial (Σ, CEE)-algebra $\mathcal{F}(\Sigma, CEE, X, R)$ freely generated by a set X of variables (generators) and a set R of Σ-equations on X where CEE is a set of conditional existence Σ-equations. Based on this concept, a fully-fledged theory

of Partial Algebraic Specifications, including free functor semantics as well as limits and colimits of signatures, specifications and partial algebras, resp., has been developed [6].

We consider here the case of total algebras. Any (Σ, CE)-algebra $\mathcal{A}$ is isomorphic to $\mathcal{F}(\Sigma, CE, A, R_{\mathcal{A}})$, which, in turn, is isomorphic to $\mathcal{F}(\Sigma, \emptyset, A, R_{\mathcal{A}})$ since $R_{\mathcal{A}}$ also encodes the fact that $\mathcal{A}$ satisfies all the conditional equations in CE. However, there may be other, hopefully finite, sets X and R such that $\mathcal{A} \cong \mathcal{F}(\Sigma, CE, X, R)$. For any such syntactic representation (X, R) of a (Σ, CE)-algebra $\mathcal{A}$ and any specification morphism $\varphi : (\Sigma, CE) \to (\Sigma', CE')$, we can construct the free algebra $\mathcal{F}_{\varphi}(\mathcal{A})$ as the Σ'-algebra $\mathcal{F}(\Sigma', CE', \varphi(X), \varphi(R))$, which is a quotient of the Σ'-term algebra $\mathcal{T}_{\Sigma'}(\varphi(X))$.

Freely generated algebras also play a crucial role in proving the completeness of the deduction calculus for conditional equations [6,7]: A deduction rule generates new equations from a set of given equations. Any conditional equation can be transformed into a deduction rule and vice versa. Given (X, R), the deduction calculus generates the smallest Σ-congruence $C(\Sigma, CE, X, R)$ in $\mathcal{T}_{\Sigma}(X)$ which contains (X, R) and is closed w.r.t. the rules arising from CE. We consider the quotient Σ-term algebra $\mathcal{F}(\Sigma, CE, X, R) = \mathcal{T}_{\Sigma}(X)/C(\Sigma, CE, X, R)$. For any Σ-algebra $\mathcal{A}$ we have $\mathcal{A} \cong \mathcal{T}_{\Sigma}(A)/ker(id^*_A)$ for the Σ-homomorphism $id^*_A : \mathcal{T}_{\Sigma}(A) \to \mathcal{A}$, and it can be shown that $\mathcal{A}$ satisfies a set CE of conditional equations if, and only if, the kernel $ker(id^*_A)$ is closed under the deduction rules arising from CE. This insures that $\mathcal{F}(\Sigma, CE, X, R)$ is a (Σ, CE)-algebra. To show the completeness of the deduction calculus, we only have to prove that $\mathcal{F}(\Sigma, CE, X, R)$ is indeed freely generated by (X, R). Note that we work here with a kind of semantic deduction theorem: A set CE of conditional equations entails a conditional equation $(X : R \to t = t')$ if, and only if, $(t, t') \in C(\Sigma, CE, X, R)$.

In the East-German school of Algebraic Specifications, we do have syntactic representations $(A, R_{\mathcal{A}})$, (X, R) of Σ-algebras which are well distinguished from signatures and algebras, respectively. At the same time, deduction means the step-wise generation of the congruence relations $C(\Sigma, CE, X, R)$ starting with (X, R). In [7], we describe these congruence relations, for example, as fixed points of so-called derivation operators describing the effect of parallel one-step applications of deduction rules. We are convinced that the definition of any specification formalism would benefit if it includes a separated "technological layer" where the syntactic representations of semantic structures live and where we can describe the effects of deduction explicitly and in detail.

Requirement 2: Define a separated technological layer where the syntactic representations of semantic structures live and where deduction takes place.

Looking back, we have been left, after all the years, with two related questions:

Question 1: Is there a general principle behind the one-to-one correspondence between conditional equations and deduction rules?

Question 2: Is there indeed a kind of general semantic deduction theorem behind the equivalence between entailment of conditional equations and entailment of equations?

We hope that our framework will enable us to give satisfactory answers.

1.1.2. Categorical Algebra

Due to Lawvere [8], one can construct for any specification (Σ, E) with E a set of Σ-equations a category $\mathsf{FP}_{(\Sigma,E)}$ with finite products such that the category of all (Σ, E)-algebras is equivalent to the category of all product preserving functors from $\mathsf{FP}_{(\Sigma,E)}$ into Set. Analogously, one can construct for any specification (Σ, CE) with CE a set of conditional Σ-equations a category $\mathsf{FL}_{(\Sigma,CE)}$ with finite limits such that the category of all (Σ, CE)-algebras is equivalent to the category of all finite limit preserving functors from $\mathsf{FL}_{(\Sigma,CE)}$ into Set.

In [9], we generalized this result to many-sorted signatures and partial algebras. We showed how to construct for any specification (Σ, CEE), with Σ a many-sorted signature and CEE a set of conditional existence Σ-equations, a category $\mathsf{FL}_{(\Sigma,CEE)}$ with finite limits such that the category of all partial (Σ, CEE)-algebras is equivalent to the category of all limit preserving functors from $\mathsf{FL}_{(\Sigma,CEE)}$ into the functor category Set^S with S the corresponding discrete category, i.e., set, of sorts declared in Σ.

The construction of those syntactic categories starts by introducing objects that correspond to declarations of finite sets of variables. After adding for each operation symbol in Σ a morphism between the appropriate objects, one continues by constructing new morphisms and an equivalence relation between morphisms. In case of finite product categories $\mathsf{FP}_{(\Sigma,E)}$, no other objects are generated while in case of the finite limit categories $\mathsf{FL}_{(\Sigma,CE)}$ and $\mathsf{FL}_{(\Sigma,CEE)}$, resp., we have to introduce new objects (X, R) representing the set of "solutions" of the set R of (existential) Σ-equations on X, i.e., a corresponding equalizer. Triggered by this example and supported by later experiences, especially with diagrammatic specification techniques, we vote for:

> Requirement 3: Define variables prior to operation and predicate symbols and use variables to define the arities of operations and predicates.

In [9], we specified finite limit categories as partial Σ_{FL}-algebras (with Σ_{FL} a signature declaring two sorts `Ob`, `Mor` and operations like `source`, `target`, `composition`, `product`, `equalizer`, `subobject`, ...) satisfying a corresponding set CEE_{FL} of conditional existence equations. The category $\mathsf{FL}_{(\Sigma,CEE)}$ was then constructed as the freely generated partial (Σ_{FL}, CEE_{FL})-algebra $\mathcal{F}(\Sigma_{FL}, CEE_{FL}, OP + CEE, R)$ with $OP + CEE$ declaring one variable of sort `Mor` for each operation symbol in Σ and one variable of sort `Mor` for each conditional existence equation in CEE. R describes source and target of the variables in $OP + CEE$ as well as the subobject property of the variables in CEE.

Thus, what we did is to reuse the formalism of partial algebras and conditional existence equations on the higher conceptual level of formalisms to coin a (meta) specification of the specification formalism "finite limits". In the process, we downgraded operations and conditional equations, playing the leading part in Section 1.1.1, to simple variables (generators). It seems quite natural to require a similar flexibility from our conceptual framework:

> Objective 1: The framework should enable us to describe and to work with specification formalisms on different conceptual levels in a uniform way.

1.1.3. Sketches in Category Theory

Categories are graphs equipped with identities and composition; thus, a string-based formalism like algebraic specifications, for example, may be not always the most adequate tool to describe and reason about categorical structures.

In the 1960s, Charles Ehresmann invented a graph-based specification formalism – the so-called sketches. Later sketches were promoted for applications in computer science by Barr and Wells [10] and applied to data modeling problems by Johnson and Rosebrugh [11] (see [12] for a survey).

A sketch $\mathbb{S} = (G, D, L, K)$ consists of a graph G and sets D, L and K of diagrams in G. In Category Theory, a diagram in a graph G of shape I is a graph homomorphism $\delta : I \to G$. A model $\mathcal{M}$ of a sketch $\mathbb{S}$ in a category C is a graph homomorphism from G to the underlying graph of C that takes every diagram in D to a commutative diagram, every diagram in L to a limit diagram and every diagram in K to a colimit diagram [10].

We use in this paper the term "diagrammatic" as a synonym for "graph-based" in a broad sense. We consider, for example, any functor (presheaf) $\mathtt{F} : \mathsf{C} \to \mathsf{Set}$ with C a small category as a "graph-based" structure. Sketches give us a diagrammatic pendant to algebraic specifications at hand and are, at the same time, more expressive. Equational specifications can be equivalently described by finite product sketches, i.e., sketches where

K is empty and L contains only finite product diagrams, while algebraic specification with conditional equations can be transformed into equivalent finite limit sketches with K empty and L containing only finite limit diagrams.

Analogous to Section 1.1.2, we can construct for any finite product sketch $\mathbb{S} = (G, D, L)$, for example, a corresponding finite product category freely generated by $\mathbb{S}$. The methodologically important observation is that the items in the graph G now play the role of "variables (generators)" while the diagrams in D and L are the "defining relations".

1.1.4. Generalized Sketches

Sketches are a very expressive specification formalism but reveal some essential deficiencies when it comes to the formalization of diagrammatic specification techniques in Software Engineering, for example (see the discussion in [13]).

We have to deal with other properties than just commutativity, limit or colimit. In addition, we meet structures that go beyond plain graphs like typed graphs or E-graphs [14], for example.

Extending the sketch formalism along the two dimensions – properties and/or structures – we arrive at generalized sketches. Generalized sketches were developed in the 1990s independently by Makkai, motivated by his work on an abstract formulation of Completeness Theorems in Logic [15], and a group in Latvia around Diskin, triggered by their work on data modeling [16,17].

To define a certain generalized sketch formalism, we chose a category Base which may differ from the category Graph of graphs. We coin for each property we want to deal with in our formalism a predicate symbol P and define, analogous to the shape graphs in traditional sketches, the arity of this predicate by an object αP in Base. Analogous to a diagram in a sketch, we define an atomic statement about an object K in Base by a morphism $\delta : \alpha P \to K$ in Base. A generalized sketch $\mathbb{K} = (K, St^{\mathbb{K}})$ is then nothing but an object K in Base together with a set $St^{\mathbb{K}}$ of atomic statements about K (see [13]).

1.1.5. Diagram Predicate Framework (DPF)

Software models and a plethora of modeling artifacts in various scientific and industrial areas are essentially diagrammatic. Traditional string-based formalisms turn out to be unwieldy and inadequate to define syntax and semantics of diagrammatic modeling techniques and to formalize diagrammatic reasoning. Instead of trying to emulate diagrammatic models and reasoning by means of traditional string-based formalisms, we adapted therefore generalized sketches when we started, around fifteen years ago, to work with Model Driven Software Engineering (MDSE).

Software models are (or, at least, appear as) graph-based structures complemented with constraints the modeled software system has to comply with. For us, it was striking that generalized sketches are the most adequate concept to formalize those artifacts. A software model can be formalized as a generalized sketch $\mathbb{K} = (K, St^{\mathbb{K}})$, where K represents the underlying graph-based structure of the model and $St^{\mathbb{K}}$ the set of constraints in the model. We further developed the generalized sketch approach as a theoretical foundation of MDSE [13,18–20] and called it, after a while, the Diagram Predicate Framework (DPF) since it turned out to be nearly impossible to convince software engineers that a "sketch" is something precise with a well-defined syntax and semantics. For the same reason, generalized sketches are called diagrammatic specifications in DPF.

DPF has been applied to a wide range of problems in MDSE with a special focus on model transformations and meta-modeling [18,21]. Thereby, we restricted ourselves to categories of graphs or typed graphs, respectively, as the base categories. (To reduce self citation, we followed the suggestion of the editors and dropped all references to papers just illustrating applications of DPF but not being relevant for the content of the paper).

While sketches and companions are relegated to a niche existence in all the traditional formalisms we discussed so far, they take center stage in DPF. The framework, presented

in this paper, arose to a big extent from the attempt to lift ideas and insights from the development of the theoretical foundations of DPF to a more general level.

> Objective 2: The framework provides a formalization of the general idea of sketches as syntactic descriptions and/or representations of semantic entities.

Our hope is that this sketch-centered approach enables us to achieve another goal.

> Objective 3: The framework allows us to describe, in a uniform way, not only string-based formalisms, like Algebraic Specifications and First-Order Logic, but also a wide variety of diagrammatic specification formalisms/techniques.

At present, DPF does have some deficiencies that we will discuss shortly.

Expressiveness of Statements

We utilize in DPF only atomic statements, called "atomic constraints", i.e., statements like $\texttt{parent}(Anna, Uwe, Gabi)$ in predicate logic stating that *Uwe* and *Gabi* are the parents of *Anna*. With those statements, we can not express all relevant constraints for software systems. String-based languages like the Object Constraint Language (OCL), for example, are traditionally used to express those constraints. OCL is built upon a fragment of first-order predicate logic and we want to extend the diagrammatic language of sketches in such a way that we can work with statements incorporating the usual logical connectives as well as universal and existential quantification. We want to be able to formulate statements like $(\exists x_1 \exists x_2 \exists x_3 : \texttt{parent}(Anna, x_2, x_3) \wedge \texttt{parent}(x_1, x_2, x_3))$ in traditional first-order predicate logic stating that *Anna* is the sibling of someone.

Our framework obtained its abstract appearance after we realized that our initial ideas to define such an extension for graph-based sketches would work in arbitrary categories!

Structure of Software Models

There are plenty of different kinds of software models. For each kind, there is a corresponding description of the required structure of software models of this kind. Those descriptions are often called meta-models. Adapting the concepts sketch-axiom [15] and graph constraint [14], we introduced "universal constraints" and "negative universal constraints", respectively, to specify the structure of software models (sketches) [18,21].

Universal constraints are, however, not expressive enough to specify all the restrictions we want or have to impose on software models. Analogous to arbitrary first-order statements, to be used as components of sketches, we want to also define therefore arbitrary first-order sketch constraints to be used to specify the structure of sketches.

To achieve this goal, we have been choosing a more unconventional approach. We neither wanted to encode traditional first-order logic of binary predicates by graphs [22] nor to emulate nested graph conditions by traditional first-order formulas [23]. We instead developed a method to define, in a conservative way, first-order constraints in arbitrary categories of sketches. By conservative, we mean that the application of our universal method to different categories of graphs, as in [22–25], for example, allows us to describe the various corresponding variants of (nested) graph constraints.

To validate the use of the term "first-order", we have to ensure, in addition, that the application of our method to the category Set results in constraints comprising essential features of traditional first-order predicate logic.

Semantics of Diagrammatic Predicates

The advantage of the traditional Ehresmann sketches in Category Theory is that there are fixed universal definitions (formulated in a language based on the concepts graph, composition and identity) of the properties commutative, limit and colimit, respectively. Since these definitions axiomatize the concepts limit and colimit "up to isomorphism", we can presuppose a fixed semantics of all corresponding diagrammatic predicates in any category, i.e., for any fixed interpretation of the concepts graph, composition and identity complying to the axioms of a category.

A price we have to pay, moving from Ehresmann sketches to generalized sketches, is that we have to describe the intended semantics of the predicates we want to include in a formalism on our own. In some cases, a complete axiomatization of the semantics of predicates will be not feasible but we should provide, at least, a partial axiomatization.

At present, we do have in DPF only the very simple notion of sketch entailment at hand to express properties of predicates. We have to extend this notion or find other notions of "arrows" between sketches that provide more appropriate tools for the axiomatization of the semantics of predicates.

On the other side, if we find a way to define arbitrary first-order diagrammatic statements, we will also have closed formulas, like $(\forall x_1 \exists x_2 \exists x_3 : \texttt{parent}(x_1, x_2, x_3))$, available for axiomatization purposes.

We intend to develop necessary tools to describe the semantics of diagrammatic predicates. We want to understand how these tools are related and, especially, find an answer to the question:

> Question 3: How are the concepts specification morphism, universal constraint and specification entailment in DPF actually related?

Operations and Substitutions

One of the crucial motivations to write this paper was to find an answer to the:

> General Question: What mathematical infrastructure we need to define a formalism enabling us to specify semantic structures with the full expressive power of first-order predicate logic?

Our answer will be: We need nothing but a category!

We are able to give such a general answer since we use only predicate symbols and no operation symbols to construct first-order statements in context. This restriction allows us to realize the translation of first-order statements along context morphisms by simple composition. In particular, there is no need for any kind of "substitution" to define those translations and thus to construct first-order formalisms for specification purposes.

For a future development of reasonable deduction calculi within our framework, we have to rely, however, on "substitutions" and, to have substitutions at hand, we need more infrastructure than just a simple category. In particular, we will need well-behaved pushout constructions as it will be shortly demonstrated in the paper.

Makkai's work [15,26] exemplifies that predicates may be, in principal, quite sufficient to build reasonable specification formalisms.

However, for applications in Software Engineering, for example, operations are sadly missed. Therefore, we are also interested in finding out if and how we can define operations in arbitrary categories. As an initial step, we started to develop a theory of graph operations [3]. It turns out that the step from traditional set operations to graph operations is not trivial at all. We are, however, optimistic that it will be possible to lift the concepts and results of a future comprehensive theory of graph operations, at least, to the level of arbitrary presheaf topoi.

Deduction

To keep software models readable and feasible, we should not overload them with unnecessary items and/or information. In particular, we should drop information that can be derived from the already given information.

To put this principle into practice, we have, however, to rely on mechanisms to derive information. Applied to DPF, this means, especially, that we need rules enabling us to deduce statements from given statements and those deduction rules should be sound.

In the paper, we introduce the concept sketch arrow and discuss the utilization of sketch arrows as deduction rules. The development of a fully fledged deduction calculus for Logics of Statements has, however, to be left as a topic of future research. We will,

nevertheless, present and discuss some vital observations, insights and ideas for this future expansion of our framework.

Category Theory can be seen as a diagrammatic specification formalism since it is based on the concepts graph, composition and identity. The development of our abstract framework is also triggered and guided by the quest to put this understanding on a precise formal ground and to develop a purely diagrammatic presentation of Category Theory where the properties commutative, limit and colimit are described by first-order statements on graphs. Most of the diagrammatic pictures in textbooks on Category Theory are nothing but sketches. The vision is to define concepts and to prove results in Category Theory based on pure "diagrammatic reasoning" – or, to formulate it differently: Let us present Category Theory in such a way that "diagram chasing" becomes a precise and well-founded proof technique. As result of such a project, one would probably end up with something very much related to the language of diagrams introduced in [27] and used in [28] to present and define categorical concepts and carry out proofs in a diagrammatic manner. We became acquainted with this language only in the final stage of writing this paper and will include a discussion of this language in the future development of a fully fledged deduction calculus for Logics of Statements.

Meta-Modeling

In DPF, we utilize categories of typed graphs, i.e., slice categories, to define the semantics of sketches. This enabled us to formalize arbitrary deep modeling hierarchies in a quite straightforward way. In this paper, we follow the tradition in logic and work with a Tarskian semantics of sketches, i.e., we work with functor categories instead of slice categories. This makes the formalization of modeling hierarchies rather involved.

Meta-modeling is a big topic on its own and, at the present stage, we are not capable of providing a detailed analysis and treatment of meta-modeling in Logics of Statements. The examples are, however, designed in such a way that we can, at least, point at the meta-modeling issue. We have to include, nevertheless, meta-modeling is an important item on our overall wish list:

Objective 4: The framework enables us to address and formalize meta-modeling.

1.1.6. Abstract Model Theory

From our various studies in Abstract Model Theory, the technical report [29] is particularly relevant for the present paper. That time, we proved in detail and in a systematic way that four specification formalisms are indeed institutions. Our main finding was that the proof of the satisfaction condition always boiled down to the existence of what we called corresponding assignments and corresponding evaluations, respectively. This finding has been integrated later by Pawlowski in his concept of context institutions [30]. One of the main motivations for context institutions was to incorporate open formulas in the abstract description of specification formalisms and the term context has been coined as an abstract pendant for a "set of variables".

What we call feature expressions in our framework are nothing but a generalization of open formulas. We differentiate, however, conceptually between variable declarations and contexts. In some specification formalisms, both concepts may denote the same thing. In other formalisms, any variable declaration will be also a context but not vice versa. In addition, there can be formalisms where variable declarations and contexts are kept apart, as in Description Logic for example. We use the term context as an abstract pendant for things like a set of generators in Group Theory, an underlying graph of a sketch in Category Theory, an underlying graph of a software model in Software Engineering, a set of literals (atomic values) in Logic Programming and a set of individual names in Description Logic, for example.

1.2. Content and Structure of the Paper

Section 2 recapitulates some basic concepts and corresponding notational conventions. We include a short discussion concerning foundations and outline how the tuple notation is used in this paper to represent (partial) finite maps.

In Section 3, we present a universal mechanism to define first-order statements and their semantics in arbitrary categories. We show that any choice of the seven parameters we are going to introduce (see Figure 1) gives us a corresponding Institution of Statements at hand. The concept of institution [2,31] is a very simple one and lives on the same abstraction level as categories and functors. We utilize institutions as a very convenient guideline to present logical formalisms in a uniform and well-organized way. The satisfaction condition is the only more complicated thing and simply tells us that we designed syntax and semantics compatible in the way that the translation of sentences corresponds exactly to model extensions (see [32–34]). Thus, validating the satisfaction condition is a kind of sanity check for the design of our formalism. At the beginning of the section, we introduce the five running examples we have chosen to illustrate and validate our definitions and constructions.

At the present stage, Institutions of Statements do not incorporate operations since we have not found yet a way to define operations in arbitrary categories. To close, nevertheless, the circle to the ideas and motivations discussed in the Introduction section 1.1.1, we recapitulate in Section 4 the traditional concepts of operations on sets and many-sorted equations. We show that the procedure we developed in Section 3 to construct Institutions of Statements enables us also to construct corresponding Institutions of Equational Statements. Substitutions play a central role in Universal Algebra, and this section may also provide some hints and guidelines for the future development of a more abstract and general account of substitutions in Logics of Statements.

Any institution gives us a corresponding category of presentations and an extension of the model functor of the institution to the category of presentations at hand [2,31]. In Section 5, we outline this construction for Institutions of Statements and Institutions of Equations, respectively. To distinguish presentations for Institutions of Statements (Equations) from presentations in general, we will use the term sketch for these specific presentations. The general theory of institutions [2,31] also provides us with a standard notion of morphism between sketches (presentations). Those morphisms are of minor importance in this paper. We introduce and investigate, in addition, sketch arrows and sketch implications as well as the relationships between these three concepts. As a pendant to elementary diagrams in traditional first-order logic, we define sketch encodings of semantic structures and will give a kind of positive answer to Question 4 (p. 39): Is there any justification to ignore completely the concept of semantic structure (model)?

To describe the syntactic structure of software models and, more generally, the structure of sketch encodings of semantic structures, we introduce and study in Section 6 arbitrary first-order sketch conditions and sketch constraints, thereby unifying and generalizing the different concepts of graph conditions and graph constraints in the area of Graph Transformations. We outline that we can, analogous to the hierarchy of generalized sketches in [15], also establish a conceptual hierarchy of sketches and sketch constraints. Moreover, we present some vital observations, insights, concepts and ideas to establish a basis for the future development of deduction calculi for Institutions of Statements.

We conclude the paper with a discussion of the results, findings and shortcomings of the paper and highlight future research directions.

The only categorical concepts we actually use in this paper are category, functor, product, functor category and slice category, and a basic understanding of these concepts is recommended. Looking up the definition of institutions may not be necessary but helpful.

2. Notations and Preliminaries

C_{Obj} 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. We use the diagrammatic notation $f;g : a \to c$ for the composition of morphisms $f : a \to b$ and $g : b \to c$ in C. $\mathsf{C} \sqsubseteq \mathsf{D}$ states that category C is a subcategory of category D. A category C is *small* if the collection C_{Mor}, and thus also the collection C_{Obj}, is a set. Cat is the category of all small categories. Set denotes the category of all sets and all (total) maps, while Par is the category of all sets and partial maps. We consider Set as a subcategory of Par. Cat, Set and Par are not small!

A (directed multi) graph $G = (G_V, G_E, sc^G, tg^G)$ is given by a collection G_V of vertices, a collection G_E of edges and maps $sc^G : G_E \to G_V$, $tg^G : G_E \to G_V$ assigning to each edge its source and target vertex, respectively. $gr(\mathsf{C})$ denotes the underlying graph of a category C, i.e., we have $gr(\mathsf{C})_V := \mathsf{C}_{Obj}$ and $gr(\mathsf{C})_E := \mathsf{C}_{Mor}$. A graph G is small if G_V and G_E are sets. A homomorphism $\varphi : G \to H$ between two graphs is given by a pair of maps $\varphi_V : G_V \to H_V$, $\varphi_E : G_E \to H_E$ such that $sc^G; \varphi_V = \varphi_E; sc^H$ and $tg^G; \varphi_V = \varphi_E; tg^H$. Graph is the category of all small graphs and all graph homomorphims between them.

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

Remark 1 (Foundations)**.** *We rely on Tarski–Grothendieck set theory, which is based on the concept of Grothendieck universes. That is, we allow ourselves to work, in principal, with open hierarchies of sets, graphs and categories, respectively. In contrast, many expositions of set theory and category theory, respectively, rely on a strict two level approach. We cite from [35], page 5:*

> *Is* CAT *a category in itself? Our answer here is to treat* CAT *as a regulative idea; which is an inevitable way of thinking about categories and functors, but not a strictly legitimate entity. (Compare the self, the universe, and God in Kant "Kritik der Reinen Vernunft".)*

Here, we work with a three-level hierarchy. That is, we also consider SET, GRAPH *and* CAT *as legitimate entities but take the level above as a "regulative level".*

In view of CAT *the category* Cat *appears in two different roles: First,* Cat *is an object in* CAT*. Second,* Cat *is a subcategory of* CAT*. We consider the inclusion functor* $\mathsf{Cat} \sqsubseteq \mathsf{CAT}$ *as an anonymous* coercion functor *which embeds any object* C *in* Cat *into the bigger context of* CAT *where we can even consider functors between* C *and* Cat*, for example. (We use the term coercion analogous to programming languages where coercion describes the implicit conversion of a value into another equivalent value of a different data type). If necessary, we will indicate in what role a small category* C *appears in a certain situation in* CAT*, namely as an object in* $\mathsf{Cat} \in \mathsf{CAT}_{Obj}$ *(the default case) or as an element in* CAT_{Obj}*, respectively.*

Analogously, we assume corresponding anonymous coercion functors $\mathsf{Set} \sqsubseteq \mathsf{SET}$ *and* $\mathsf{Graph} \sqsubseteq \mathsf{GRAPH}$*, respectively. Note that the isomorphisms between small categories* C *and the corresponding objects in* CAT *as well as the anonymous coercion functors are not living in* CAT*! They are located on our third regulative level. Finally, we assume also implicit coercion from the categories* SET, GRAPH *and* CAT*, respectively, to the regulative level.*

In other words, we comply with the following principles: (1) Any item on a certain level of the hierarchy can be used at any level above but it can not be used at any level below the level where it has been declared or constructed. (2) Located on a certain level of the hierarchy, we can see, declare and construct items on this level and on all levels below. (3) We are, however, not allowed to push an item to a lower level! Instead, we have to declare or construct a "new" item on the lower level and establish an isomorphism between the given item and the new item. The lowest level, where the isomorphism could be established, is the level of the given item but sometimes we will be only able to establish the isomorphism on an even higher level.

To achieve Objective 3 (p. 6), we have to pay a small price. In addition to the conventional interpretation of an n-tuple $(a_1, \ldots, a_n)$ as a "list of values of length n", we will also work with a more unconventional interpretation. We interpret an n-tuple $\mathbf{a} = (a_1, \ldots, a_n)$ with $n \geq 1$ and $a_1, \ldots, a_n \in A$ as a convenient shorthand notation for an "indexed array" of

length n, i.e., for a set of assignments $\{1 \mapsto a_1, \ldots, n \mapsto a_n\}$ representing a map $\mathbf{a} : [n] \to A$ with $[n] := \{1, \ldots, n\}$ and $\mathbf{a}(i) = a_i$. That is, the numbers in $[n]$ indicate the corresponding *position* in the tuple. The empty tuple $()$ represents, in such a way, the only map from $[0] := \emptyset$ into A.

Given an $[n]$-indexed family $A_1, \ldots, A_n$, $n \geq 1$ of sets, i.e., a map $A : [n] \to \mathsf{Set}_{Obj}$, we denote the set of all maps $\mathbf{a} : [n] \to \bigcup_{i \in [n]} A_i$ with $\mathbf{a}(i) \in A_i$ for all $i \in [n]$ by $A_1 \otimes \ldots \otimes A_n$, $\bigotimes_{i \in [n]} A_i$ or simply $\bigotimes A$, respectively. Relying on the assumption that $[n]$ is (implicitly) equipped with the total irreflexive order $1 < 2 < \ldots < n$, we can still use the traditional tuple notation to represent those maps, as discussed in the last paragraph. The traditional Cartesian product $A_1 \times \ldots \times A_n$ and $\bigotimes A$ are isomorphic and both give us a categorical product of the family $A_1, \ldots, A_n$ of objects in Set at hand. If necessary, we will use the term traditional tuple to indicate the traditional interpretation of a tuple as a simple "list of values".

To describe, for example, the concept of a row in Relational Databases (see Section 3.1.5), we also take the step from indexed arrays to "associative arrays".

Instead of the standard sets $[n]$ of indexes, we consider arbitrary finite sets I of indexes (identifiers, names) with n elements. For an I-indexed set $A = (A(i) \mid i \in I)$, i.e., a map $A : I \to \mathsf{Set}_{Obj}$, we denote by $\bigotimes_{i \in I} A(i)$, or simply $\bigotimes A$, the set of all maps $\mathbf{a} : I \to \bigcup_{i \in I} A(i)$ with $\mathbf{a}(i) \in A(i)$ for all $i \in I$. $\bigotimes A$ is a categorical product of the I-indexed family A of objects in Set where for any $i \in I$ the corresponding *projection map* $\pi_i : \bigotimes A \to A(i)$ is simply defined by $\pi_i(\mathbf{a}) := \mathbf{a}(i)$ for all $\mathbf{a} \in \bigotimes A$.

Each element $\mathbf{a}$ in $\bigotimes A$ can be represented by a corresponding *associative array*, i.e., by the set $\{i \mapsto \mathbf{a}(i) \mid i \in I\}$ of assignments. To be able, however, to utilize the tuple notation to represent the elements in $\bigotimes A$, we have to equip the set I, explicitly, with a fixed (!) total order $i_1 < i_2 < \ldots < i_n$. Under this assumption, we can then represent each $\mathbf{a}$ in $\bigotimes A$ by the tuple $(a_1, \ldots, a_n)$ with $a_j = \mathbf{a}(i_j)$ for all $1 \leq j \leq n$.

In practice, it is often more convenient to work with interpretation categories instead of functor categories. An interpretation of a graph G in a category C is a graph homomorphism φ from G to $gr(\mathsf{C})$ denoted by $\varphi : G \to \mathsf{C}$. A natural transformation $\mu : \varphi \Rightarrow \psi$ between two interpretations $\varphi : G \to \mathsf{C}$ and $\psi : G \to \mathsf{C}$ is a family $\mu_v : \varphi_V(v) \to \psi_V(v)$, $v \in G_V$ of morphism in C such that $\varphi_E(f); \mu_u = \mu_v; \psi_E(f)$ for all edges $f : v \to u$ in G. All interpretations of G in C and all natural transformations between them constitute the interpretation category $[G \to \mathsf{C}]$ with composition – the vertical composition of natural transformations. (In [10], interpretations $\varphi : G \to \mathsf{C}$ are called "models of G in C", and the notation $\mathbf{Mod}(G, \mathsf{C})$ is used instead of $[G \to \mathsf{C}]$. For our purposes, the more neutral and general term "interpretation" is more convenient, and we do not want to overload the term "model" too heavily). For convenience and uniformity reasons, we will often consider a set A as a graph without edges and use the interpretation category $[A \to \mathsf{C}]$ to represent all maps from A into C_{Obj}. Moreover, we will also use the more compact notations C^G and C^A instead of $[G \to \mathsf{C}]$ or $[A \to \mathsf{C}]$, respectively.

3. Institutions of Statements

Before we are going to define Institutions of Statements, we outline the running examples we have chosen to illustrate and validate our definitions. The reader should be aware that our framework is very abstract and thus also very flexible. It enables us to present one and the same specification formalism in various ways. Thus, the way we have chosen for each single sample formalism may be not the most adequate one and, especially, not the one preferred by the reader.

3.1. Examples

3.1.1. First-Order Predicate Logic (FOL)

Our category independent framework does not incorporate operations. Therefore, we examine many-sorted first-order predicate logic without functions. We consider many-sorted signatures $\Sigma = (S, P, ar : P \to S^*)$ with S a set of sort symbols, P a set of predicate

symbols and a map $ar : P \to S^*$ assigning to each predicate symbol its arity. We may sometimes omit the word 'symbol' and simply refer to sort symbols as sorts and to predicate symbols as predicates. We show that any many-sorted signature can be represented quite naturally within our framework and therefore gives rise to different institutions of statements. We will demonstrate this by means of a sample signature.

3.1.2. Description Logic (ALC)

Description logics are a family of formal knowledge representation languages. We discuss the prototypical description logic Attributive Concept Language with Complements (ALC) which can be seen as a fragment of unsorted FOL without functions (see [36]). We include this non-classical example to illustrate that our framework may be indeed suitable to describe a wide variety of specification formalisms.

This adaption of First-Order Logic to deal with the practical problem of knowledge representation and the example of DPF demonstrate that contexts and sketches, as they are defined in our framework, appear quite natural as conceptual building blocks in practical specification formalisms.

3.1.3. The Formalism "First-Order Predicate Logic" (mFOL)

This example is meant to provide some evidence that our framework lives up to Objective 1 (p. 4). In the FOL-example, we work within the formalism many-sorted first-order logic without functions. Here, we move one abstraction level up and intend to describe this formalism as such. The "m" in "mFOL" stands for meta.

The sketches in the FOL-example are related to concepts like generators and defining relations in Group Theory and literals and facts in Logic Programming but are not a common ingredient in traditional expositions of First-Order Logic. The sketches that appear in this example, however, reconstruct the concept many-sorted signature as we meet it in the FOL-example. As an example, we reconstruct the sample signature we will work with in the FOL-example. Thus, the FOL-example and the mFOL-example together exemplify the topic of meta-modeling.

3.1.4. Category Theory (CT)

Together with the DPF-example, this example should demonstrate the potential of our framework to support a shift of paradigm from string-based to diagrammatic specification formalisms.

Located on the same abstraction (modeling) level as the examples FOL and ALC and reflecting the viewpoint that a category is a graph equipped with composition and identities, we outline a diagrammatic version of the theory of small categories.

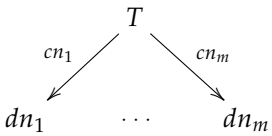
In fact, we take a step back from Ehresmann's sketches. We restrict ourselves to the language of graphs, composition and identities and reconstruct the concepts commutative diagram, limit and colimit, respectively, by means of diagrammatic first-order statements formulated in this restricted language. The universal properties defining the different kinds of limits and colimits, respectively, do have a uniform and very simple logical structure; thus, we need only a very restricted form of first-order statements to express them. In the light of this observation, our envisioned diagrammatic version of the theory of small categories goes beyond Ehresmann's sketches in the sense that we allow for utilizing arbitrary first-order statements. Even if we do not need the full "first-order power" to define limits and colimits, this power will be probably useful (or even necessary) to formulate category theoretic statements and to prove them.

3.1.5. Diagram Predicate Framework (DPF)

Now, we arrive indeed at generalized sketches since we will utilize typed graphs instead of just plain graphs as in the CT-example. We are on the same abstraction level as the mFOL-example.

DPF has been developed to describe and relate, in a uniform and precise formal way, a wide variety of diagrammatic modeling techniques in Software Engineering. Each diagrammatic modeling technique, like database schemata, ER diagrams, class diagrams, workflow diagrams, for example, is characterized by a certain footprint. A sketch for such a footprint formalizes then nothing but a single software model. As an example, we outline in this paper a revised and extended version of our diagrammatic Relational Data Model (RM) [18,21].

In Relational Databases, we do have data types and tables with rows and columns. In addition, we can declare different kinds of constraints. A table is identified by a name, and each table has a fixed non-empty set of columns. All columns in a certain table are identified by a unique name; thus, the order of columns is immaterial. It is allowed to use the same column name in different tables. All values in a certain column have to be of the same data type. A table is considered as a set of rows with one cell for each column. In some cells of a table, there may be no values. A row with no values at all is not allowed! Let us declare a table with name T, a corresponding set $C = \{cn_1, \ldots, cn_m\}$ of column names and a declaration of a data type name dn_j for each column name cn_j.

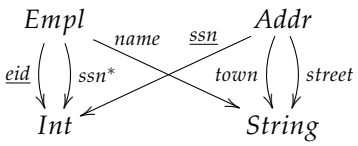


We represent this declaration by the graph shown above. To define the semantics of table T, we first have to fix the semantics of the data type names dn_j by assigning to each data type name dn_j a fixed set D_{dn_j} of data values. This gives us an C-indexed set $D = (D_{dn_j} \mid cn_j \in C)$ at hand.

Since there may be no values in some of the cells in a row, we generalize the definitions in Section 2 and describe a row $\mathbf{r}$ in table T as a partial map $\mathbf{r} : C \nrightarrow \bigcup D$ with $\mathbf{r}(cn_j) \in D_{dn_j}$ as long as $\mathbf{r}(cn_j)$ is defined. We denote by $\bigotimes^p_{j \in I} D_{dn_j}$, or simply $\otimes^p D$, the set of all those partial maps except the completely undefined map (empty row). For any $cn_j \in C$, we obtain as projection a partial map $\pi_{cn_j} : \bigotimes^p D \nrightarrow D_{dn_j}$ defined for all $\mathbf{r} \in \bigotimes^p D$ by $\pi_{cn_j}(\mathbf{r}) := \mathbf{r}(cn_j)$ if $\mathbf{r}(cn_j)$ is defined. These projections turn $\bigotimes^p D$ into a categorical product of the C-indexed set $D = (D_{dn_j} \mid cn_j \in C)$ in the category Par of all sets and partial maps.

Reflecting the idea of a row in a table, we can still utilize the tuple notation, discussed in Section 2, to denote the elements in $\bigotimes^p D$. We fix a total order $cn_1 < cn_2 < \ldots < cn_n$ on C and represent a partial map $\mathbf{r} : C \nrightarrow \bigcup D$ by the tuple $(r_1, \ldots, r_n)$ with $r_j = \mathbf{r}(cn_j)$ if $\mathbf{r}(cn_j)$ is defined and r_j an anonymous indicator "␣" for *nothing* in all other cases.

The content of table T may change. At any point in time, however, the content (semantics) of table T is a finite subset of $\bigotimes^p D$ and the semantics of the edges cn_j are the corresponding restrictions of the projections $\pi_{cn_j} : \bigotimes^p D \nrightarrow D_{dn_j}$.



To discuss constraints, let us consider a database schema declaring two data types Int(eger), String and two tables Empl(oyee), Addr(ess) with columns as depicted in the diagram above.

Since a table is a set (!) of rows, we need a mechanism to identify rows uniquely. These are the so-called primary keys (pk). For each table, one of the columns has to be declared as a primary key. In the example, we declare the primary keys *eid* (employee identity) in table *Empl* and *ssn* (social security number) in table *Addr* indicated by underlined names. All values in a primary key have to be distinct and empty cells are not allowed. This means that the corresponding projection has to be injective and total. To require only injectivity, we declare a unique constraint and a not null constraint will enforce a total projection. We

may put both constraints on the column *ssn* in *Empl*. This will, however, not turn *ssn* into a primary key but only into a candidate key. A primary key is the one of the candidate keys we have chosen to serve as a primary key!

To store and retrieve information, the tables in a database have to be somehow connected. To find, for example, the address of an employee, we have to consult Table *Addr*. *Foreign key (fk)* constraints are the mechanism to connect tables. In the example, we declare a foreign key from column *ssn* in *Empl* to column *ssn* in *Addr* indicated by a star ssn^*. A column declared as a foreign key may contain empty cells but any value appearing in this column has to also appear in the column the key refers to. This means, especially, that both columns are required to have the same data type!

The Blueprint for Constructing Institutions of Statements

In the following subsections, we define Institutions of Statements (IoS). Each Institution of Statements is characterized by seven parameters that we will introduce step by step. The reader can keep track of the development consulting the scheme in Figure 1.

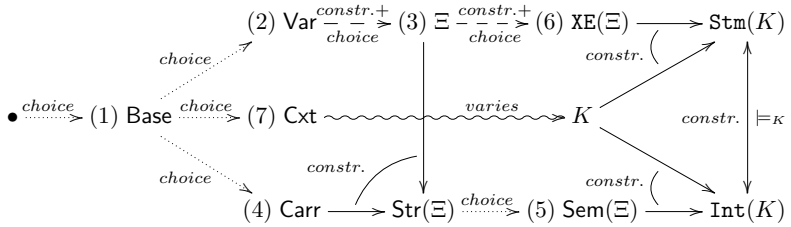


Figure 1. Stepwise construction of an Institution of Statements (IoS).

3.2. Base Category

To define a certain Institution of Statements, we first have to choose a base category comprising as well the basic syntactic entities as the semantic domains. The base category fixes, somehow, the linguistic and conceptual universe we intend to work within.

Definition 1 (First parameter: Base Category)**.** *The first parameter of an Institution of Statements is a chosen* ***base category*** Base*.*

Remark 2 (Uniformity)**.** *All components of a logic are related with each other; thus, it seems to be natural to require that they all live in the same category (universe). It turns out, however, that this uniformity requirement is not as trivial, as it looks like at a first glance. Some effort may be needed to present known logic's and formalisms in such a uniform way. One underlying cause for this kind of additional effort is that we follow the tradition in logic and work here with the semantics-as-interpretation paradigm (Tarskian semantics) (see Section 3.3.2).*

One problem could be that we become obliged to reflect a distinction between different syntactic entities, like predicate symbols and sort symbols, for example, already on the semantic level. In the more practical relational data model example, this is quite appropriate while it needs getting used to in the mFOL-example.

In other words: We take the chance to present the mFOL-example in a more unconventional way also with the intention to illustrate the flexibility of our framework.

Example 1 (FOL: Base category)**.** *There is a particular dependency in many-sorted first-order signatures. We have first to establish a set of sort symbols before we can define the arity of predicates (compare Chapter 12 in [2]). We fix a set* $S \in \mathsf{Set}_{Obj}$ *of sort symbols and consider S as a graph (without edges) in* Graph*. Relying on the coercion* $\mathsf{Graph} \sqsubseteq \mathsf{GRAPH}$ *(see Remark 1), we choose the interpretation category* $\mathsf{Set}^S = [S \to \mathsf{Set}]$ *in* CAT *as base category* Base_{FOL}*. We call the objects in* $[S \to \mathsf{Set}]$ *S-sets and the morphisms S-maps, respectively.*

Example 2 (ALC: Base category)**.** *The prototypical description logic* Attributive Concept Language with Complements (ALC) *can be seen as a fragment of unsorted FOL without functions [36], thus the category* $\mathsf{Base}_{ALC} := \mathsf{Set}$ *in* CAT *is our base category of choice.*

Example 3 (mFOL: Base category)**.** *In this example, we describe the traditional formalism* many-sorted first-order logic without functions *as such.*

In this formalism, sorts *and* predicates *are the only concepts. Therefore, we use the set* $M_{mF} := \{\boldsymbol{S}, \boldsymbol{P}\} \in \mathsf{Set}_{Obj}$ *of* concept symbols*. Relying on coercion* $\mathsf{Set} \sqsubseteq \mathsf{SET}$*, we define the base category as a slice category* $\mathsf{Base}_{mF} := \mathsf{SET}/M_{mF}$*. (For any category* C *and any object* T *in* C *we can define the slice category* C/T *with objects all pairs* (A, φ) *of an object* A *and a morphism* $\varphi : A \to T$ *in* C *and morphisms* $f : (A, \varphi) \to (B, \psi)$ *given by morphisms* $f : A \to B$ *in* C *satisfying the condition* $\varphi = f; \psi$*.) We use the term* M_{mF}-typed set *for the objects in* SET/M_{mF}*, i.e., for pairs* (A, τ_A) *of a set* A *and a* typing map $\tau_A : A \to M_{mF}$*.*

Example 4 (CT: Base category)**.** *Located on the same abstraction (modeling) level as the examples FOL and ALC and reflecting the viewpoint that a category is a graph equipped with composition and identities, we outline a diagrammatic version of the theory of small categories; thus, the category* Graph *of small graphs is chosen as the base category* Base_{CT}*. Note that* Graph *is isomorphic to the interpretation category* $[M_{CT} \to \mathsf{Set}]$ *with* M_{CT} *the graph* $E \overset{sc}{\underset{tg}{\rightrightarrows}} V$ *thus we follow somehow the same pattern as in example FOL.*

Example 5 (RM: Base category)**.** *The relational data model relies on the concepts* table, column *and* datatype*. Analogously to example mFOL, we use a graph* $M_{RM} := (\ \boldsymbol{T} \xrightarrow{\boldsymbol{C}} \boldsymbol{D}\ ) \in \mathsf{Graph}_{Obj}$ *of concept symbols and define the base category as a slice category* $\mathsf{Base}_{RM} := \mathsf{GRAPH}/M_{RM}$ *relying on the coercion* $\mathsf{Graph} \sqsubseteq \mathsf{GRAPH}$*. We use the term* M_{RM}-typed graph *for the objects in* GRAPH/M_{RM}*, i.e., for pairs* (G, τ_G) *of a graph* G *and a* typing morphism $\tau_G : G \to M_{RM}$*.*

3.3. Variables, Features and Footprints

3.3.1. Variables, Features and Footprints: Syntax

Traditionally, the construction of syntactic entities in logic, like terms, expressions and formulas, starts by declaring what variables can be used in the language of a certain logic. Often, we assume an enumerable set of variables and then any term, expression or formula is based upon a chosen finite subset of this enumerable set of variables. Moreover, variable translations can be described by maps between finite sets of variables. Generalizing this traditional approach, we announce what kind of variables we want to use in our institution.

Definition 2 (Second parameter: Variables)**.** *As the second parameter of an Institution of Statements, we choose a subcategory* Var *of the base category* Base*. We refer to the objects in* Var *as* ***variable declarations*** *while the morphisms in* Var *will be called* ***variable translations****.*

If Base *has initial objects, we assume that* Var *contains exactly one of them denoted by* $\mathbf{0}$*.*

This is a completely different view on variables compared to the tradition in the theory of institutions [2], where variables generally depend on the notion of signature.

Example 6 (FOL: Variables)**.** *Variable declarations are traditionally just finite S-sets of variables. We take* Var_{FOL} *to be the full subcategory of* $\mathsf{Base}_{FOL} = [S \to \mathsf{Set}]$ *given by all finite and disjoint S-sets* $X = (X_s \mid s \in S)$ *with* X_s *a subset of the set* $\{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots\}$ *for all* $s \in S$*.*

Example 7 (ALC: Variables)**.** *Officially, there are no variables in ALC. To describe ALC as a fragment of FOL we need, however, variables. As* Var_{ALC}*, we take the subcategory of* Set *with objects all finite subsets of the set* $\{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots\}$ *and morphisms all injective maps.*

Example 8 (mFOL: Variables). *We choose as* Var_{mF} *the full subcategory of* $\mathsf{Set}/M_{mF} \sqsubseteq \mathsf{Base}_{mF} = \mathsf{SET}/M_{mF}$ *given by all finite* M_{mF}*-typed sets* $(X, \tau_X : X \to M_{mF})$ *such that the pre-image* $\tau_X^{-1}(\boldsymbol{S})$ *is a subset of the set* $\{xs, xs_1, xs_2, \ldots, ys, ys_1, ys_2, \ldots\}$ *and* $\tau_X^{-1}(\boldsymbol{P}) = \{xp\}$.

Example 9 (CT: Variables). *The variable declarations are* graphs of variables, *i.e., we work with two kinds of variables:* vertex variables *and* edge variables *that are connecting vertex variables. We choose* Var_{CT} *to be the full subcategory of* $\mathsf{Base}_{CT} = \mathsf{Graph}$ *given by all finite graphs* $X = (X_V, X_E, sc^X, tg^X)$ *with* X_V *a finite subset of the set* $\{xv, xv_1, xv_2, \ldots, yv, yv_1, yv_2, \ldots\}$ *and* X_E *a finite subset of the set* $\{xe, xe_1, xe_2, \ldots, ye, ye_1, ye_2, \ldots\}$. *"e" stands for edge while "v" refers to vertex.*

Example 10 (RM: Variables). *As* Var_{RM}, *we choose the full subcategory of* $\mathsf{Graph}/M_{RM} \sqsubseteq \mathsf{Base}_{RM} = \mathsf{GRAPH}/M_{RM}$ *given by all finite* M_{RM}*-typed graphs* $(X, \tau_X : X \to M_{RM})$ *such that the pre-image* $\tau_X^{-1}(\boldsymbol{T})$ *is a finite subset of the set* $\{xt, xt_1, xt_2, \ldots, yt, yt_1, yt_2, \ldots\}$, $\tau_X^{-1}(\boldsymbol{D})$ *is a finite subset of* $\{xd, xd1, xd_2, \ldots, yd, yd_1, yd_2, \ldots\}$ *and* $\tau_X^{-1}(\boldsymbol{c})$ *is a finite subset of* $\{xc, xc_1, xc_2, \ldots, yc, yc_1, yc_2, \ldots\}$, *respectively.*

Guided by Requirement 3 (p. 4), we introduced variables first and can utilize them now to define arities.

Definition 3 (Third parameter: Footprint). *The third parameter of an Institution of Statements is a* ***footprint*** $\Xi = (\Phi, \alpha)$ *over* Var *given by a set* Φ *of* ***feature symbols*** *and a map* $\alpha : \Phi \to \mathsf{Var}_{Obj}$. *For any feature symbol* $F \in \Phi$, *the variable declaration* $\alpha(F)$ *is called the* ***arity of*** F. *We will often write* αF *for* $\alpha(F)$.

Remark 3 (Terminology: Footprint vs. signature). *In most of our applications of DPF, footprints occur as meta-signatures, in the sense that each specification formalism (modeling technique) is characterized by a certain footprint. Each of the formalisms Universal Algebra, Category Theory, First-Order Logic, ER diagrams, class diagrams is characterized by a certain footprint. The sketch data model in [11] corresponds to a certain footprint and so on. For the footprint of the modeling technique* class diagrams, *we refer to [18,21].*

Until today, we used in all our DPF papers the terms signature instead of footprint and predicate symbol instead of feature symbol (compare [13,21]). This turned out to be a source for serious misunderstandings and misleading perceptions; thus, we decided to coin new terms.

Remark 4 (Dependencies between features). *Extending Makkai's approach [15], we worked in [13] with categories* $\boldsymbol{\Phi}$ *of feature symbols, instead of just sets of feature symbols, and with arity functors* $\alpha : \boldsymbol{\Phi} \to \mathsf{Var}$, *instead of just arity maps. Arrows between feature symbols represent dependencies between features. This allows us to reflect, already on the level of feature symbols and thus prior to arities and semantics of features that certain features depend on (are based upon) other features. As examples, one may express that both concepts pullback and pushout are based upon the concept commutative square and that the categorical concept inverse image depends on the concept monomorphism.*

Any semantics of feature symbols then has to respect those dependencies. Dependency arrows are a tool to represent knowledge about and requirements on features prior to and independent of any kind of logic. Dependency arrows somehow make the framework of generalized sketches conceptual and structural round.

It may be worth mentioning that the concept of order-sorted algebra *is somehow related to our idea of dependencies since it works with arrows between sort symbols [37].*

In this first paper about Logics of Statements, we drop dependency arrows due to, at least, three reasons: (1) We do not want to deviate too much from the traditional first-order logic setting. (2) Dependencies trigger an additional theoretical overhead that may be not worth it at the moment. If we introduce dependencies between feature symbols, we should consequently describe, for example, to what extent and how they generate dependencies between feature expressions (introduced in

Section 3.4). On one side, this is technically not fully trivial, if possible at all. On the other side, such an effort has no relevance for our applications. (3) The requirements expressed by dependency arrows can be mimicked by the logical tools we are going to introduce later.

Example 11 (FOL: Footprint)**.** *We show, first, how an arbitrary traditional many-sorted signature* $\Sigma = (S, P, ar : P \to S^*)$ *can be transformed into a footprint* $\Xi_\Sigma = (\Phi_\Sigma, \alpha_\Sigma)$ *and then we present a sample FOL-footprint to be used in the forthcoming parts of this example.*

The set S of sort symbols has been already transformed into the IoS-setting by choosing the base category $[S \to \mathsf{Set}]$*. The set* Φ_Σ *of feature symbols is nothing but the set P of predicate symbols. Thus, it remains to transform each* $w \in S^*$ *into a corresponding S-set* X^w *of variables.*

The empty sequence $\varepsilon \in S^*$ *is simply transformed into the empty S-set* $X^\varepsilon := (\emptyset \mid s \in S)$*. A non-empty sequence* $w = s_1 s_2 \dots s_n$ *gives rise to a list* $[x_1{:}\, s_1, x_2{:}\, s_2, \dots, x_n{:}\, s_n]$ *of variable declarations, i.e., to a canonical set* $\{x_1, x_2, \dots, x_n\}$ *of variables, equipped with a canonical total order* $x_1 < x_2 < \dots < x_n$*, together with a map from* $\{x_1, x_2, \dots, x_n\}$ *into S.* X^w *is defined by* $X^w_s := \{x_i \mid s_i = s\}$ *for all* $s \in S$*. In the examples, we will use lists of variable declarations to represent S-sets of variables.*

To complete the definition of Ξ_Σ*, we simply set* $\alpha_\Sigma(\mathtt{p}) := X^{ar(\mathtt{p})}$ *for all* $\mathtt{p} \in \Phi_\Sigma = P$*.*

As an example for a FOL-footprint, we chose $S = \{prs, nat\}$ *with sort symbols "prs" for person and "nat" for natural number, respectively. The sample footprint* $\Xi_{FOL} = (\Phi_{FOL}, \alpha_{FOL})$ *is then defined by the feature symbols* $\Phi_{FOL} := \{\mathtt{parent}, \mathtt{male}, \mathtt{age}, \mathtt{less}\}$ *with the following S-sets as arities:* $\alpha_{FOL}(\mathtt{parent}) := (\{x_1, x_2, x_3\}, \emptyset)$ *represented by* $[x_1{:}\, prs, x_2{:}\, prs, x_3{:}\, prs]$*,* $\alpha_{FOL}(\mathtt{male}) := (\{x\}, \emptyset)$ *represented by* $[x{:}\, prs]$*,* $\alpha_{FOL}(\mathtt{age}) := (\{x_1\}, \{x_2\})$ *represented by* $[x_1{:}\, prs, x_2{:}\, nat]$ *and* $\alpha_{FOL}(\mathtt{less}) := (\emptyset, \{x_1, x_2\})$ *represented by* $[x_1{:}\, nat, x_2{:}\, nat]$*.*

Example 12 (ALC: Footprint)**.** *A signature in ALC declares a set* N_C *of* concept names *and a disjoint set* N_R *of* role names*. In view of Definition 3, this means defining a footprint* $\Xi_{ALC} = (\Phi_{ALC}, \alpha_{ALC})$ *with* $\Phi_{ALC} = N_C \cup N_R$*,* $\alpha_{ALC}(F) = \{x\}$ *for all* $F \in N_C$ *and* $\alpha_{ALC}(F) = \{x_1, x_2\}$ *for all* $F \in N_R$*.*

A signature in ALC also declares a set N_O *of* individual names (nominals, objects)*. In our framework, those sets of individual names are considered as* contexts *(see Definition 11).*

Example 13 (mFOL: Footprint)**.** *An mFOL-footprint describes which one of the enumerable many formal tools n-*ary many-sorted predicates *we will have at hand. As an example, we consider an mFOL-footprint* $\Xi_{mF} = (\Phi_{mF}, \alpha_{mF})$ *providing the formal tools unary many-sorted predicates, binary many-sorted predicates and tertiary many-sorted predicates, respectively.*

We have $\Phi_{mF} := \{\mathtt{un}, \mathtt{bin}, \mathtt{trt}\}$ *with* $\alpha_{mF}(\mathtt{un}) = (X_{\mathtt{un}}, \tau_{X_{\mathtt{un}}} : X_{\mathtt{un}} \to M_{mF})$ *given by* $X_{\mathtt{un}} := \{xp, xs\}$ *and* $\tau_{X_{\mathtt{un}}}(xp) := \boldsymbol{P}$*,* $\tau_{X_{\mathtt{un}}}(xs) := \boldsymbol{S}$*. Analogously to Example 11, we represent the* M_{mF}*-typed set* $(X_{\mathtt{un}}, \tau_{X_{\mathtt{un}}})$ *by the list* $[xp{:}\, \boldsymbol{P}, xs{:}\, \boldsymbol{S}]$ *of variable declarations.* $(X_{\mathtt{bin}}, \tau_{X_{\mathtt{bin}}})$ *is defined by* $[xp{:}\, \boldsymbol{P}, xs_1{:}\, \boldsymbol{S}, xs_2{:}\, \boldsymbol{S}]$ *while* $(X_{\mathtt{trt}}, \tau_{X_{\mathtt{trt}}})$ *is given by* $[xp{:}\, \boldsymbol{P}, xs_1{:}\, \boldsymbol{S}, xs_2{:}\, \boldsymbol{S}, xs_3{:}\, \boldsymbol{S}]$*, respectively. Keep in mind that, for any set M in* Set *the interpretation category* $[M \to \mathsf{Set}]$ *and the slice category* Set/M *are equivalent.*

Example 14 (CT: Footprint)**.** *Category Theory relies on a language based upon the concepts* object (vertex)*,* morphism (arrow, edge)*,* composition *and* identity*. The concept* graph *comprises already the concepts object (vertex) and morphism (arrow, edge); thus, a footprint for our diagrammatic reconstruction of the theory of small categories only needs to take care of composition and identity.*

We do not have operations in footprints; thus, we have to formalize composition and identity by means of features (predicates). Therefore, the footprint Ξ_{CT} *for the formalism Category Theory declares two feature symbols* `cmp` *and* `id`*. The arities of the feature symbols in* $\Phi_{CT} := \{\mathtt{cmp}, \mathtt{id}\}$ *are described in Table 1:*

Table 1. CT Footprint.

F	Arity $\alpha_{CT}(F)$	F	Arity $\alpha_{CT}(F)$
`cmp`	$xv_1 \xrightarrow{xe_1} xv_2 \xrightarrow{xe_2} xv_3$, $xv_1 \xrightarrow{xe_3} xv_3$	`id`	$xv \circlearrowright xe$

Example 15 (RM: Footprint). *The footprint* $\Xi_{RM} = (\Phi_{RM}, \alpha_{RM})$ *declares features* $\Phi_{RM} := \{\texttt{tb(n)}, \texttt{pk}, \texttt{fk}, \texttt{tot}, \texttt{inj}\}$ *for the concepts* table with n columns, primary key, foreign key, not null *(total) and* unique *(injective), respectively. We discussed these concepts in Section 3.1.5 where we introduced the relational data model example. The arities of the feature symbols in* Φ_{RM} *are* M_{RM}*-typed graphs* (G, τ_G) *and are described in Table 2. Analogously to Example 13, we use the colon-notation "_:_" to represent the typing morphisms* $\tau_G : G \to M_{RM}$.

Table 2. RM Footprint.

F	Arity $\alpha_{RM}(F)$
`tb(n)`	$xt\colon \mathbf{T}$ with $xc_1\colon \mathbf{c}$, $xc_n\colon \mathbf{c}$ to $xd_1\colon \mathbf{D}$ • • • $xd_n\colon \mathbf{D}$
`fk`	$xt_1\colon \mathbf{T} \xrightarrow{xc_1\colon \mathbf{c}} xd\colon \mathbf{D} \xleftarrow{xc_2\colon \mathbf{c}} xt_2\colon \mathbf{T}$
`pk`	$xt\colon \mathbf{T} \xrightarrow{xc\colon \mathbf{c}} xd\colon \mathbf{D}$
`inj`	$xt\colon \mathbf{T} \xrightarrow{xc\colon \mathbf{c}} xd\colon \mathbf{D}$
`tot`	$xt\colon \mathbf{T} \xrightarrow{xc\colon \mathbf{c}} xd\colon \mathbf{D}$

Remark 5 (Category of footprints). *We indicated the arrow from (2) to (3) in Figure 1 as* construction+choice *since we could straightforwardly define a category of footprints on* Var *while we decided to consider only one footprint. To also explore categories of footprints goes simply beyond the scope of this first paper on Logics of Statements. In Remark 20, we will, however, outline, what has to be done if we want or need to work with a category of footprints.*

3.3.2. Variables, Features and Footprints: Semantics

To make things not too complicated and to deviate not too far from traditional logic, we work here with the semantics-as-interpretation paradigm, also called indexed or Tarskian semantics. In contrast, we spelled out in [13] the semantics-as-instance paradigm, also called fibred semantics. To define the semantics of variables and features, we first have to decide for (potential) carriers of structures.

Definition 4 (Fourth parameter: Carriers). *As the fourth parameter of an Institution of Statements, we choose a subcategory* Carr *of* Base *of (potential)* ***carriers*** *of* Ξ*-structures.*

Example 16 (FOL: Carriers). *In this example, we follow the traditional approach and choose simply* $\mathsf{Carr}_{FOL} := \mathsf{Base}_{FOL} = \mathsf{Set}^S = [S \to \mathsf{Set}]$.

Example 17 (ALC: Carriers). *ALC considers only non-empty sets as potential carriers and calls them* domains (of an interpretation). *Thus, we take as* Carr_{ALC} *the full subcategory of* $\mathsf{Base}_{ALC} = \mathsf{Set}$ *given by all non-empty sets.*

Example 18 (mFOL: Carriers). *A potential carrier of a* Ξ_{mF}*-structure should provide a family of sets to define the semantics of sort symbols as well as a family of sets to define the semantics of*

predicate symbols. As Carr_{mF}*, we choose therefore the full subcategory of* $\mathsf{Base}_{mF} = \mathsf{SET}/M_{mF}$ *given by all* M_{mF}*-typed sets* (C, τ_C) *with* $C \subseteq \mathsf{Set}_{Obj}$*. Note that we consider here* Set_{Obj} *(and thus also C) as an element in* SET_{Obj} *and not as a subset of* SET_{Obj} *(compare Remark 1).*

Example 19 (CT: Carriers)**.** *We could choose only those graphs that appear as underlying graphs of small categories. We will, however, not restrict ourselves and choose, analogous to Example 16,* $\mathsf{Carr}_{CT} := \mathsf{Base}_{CT} = \mathsf{Graph}$*.*

Example 20 (RM: Carriers)**.** *Tables are sets of rows and data types are sets of data values while columns can be formalized as maps assigning to each row in a table the value in the corresponding column. As discussed in Section 3.1.5, these maps can be partial since there may be no values in some cells of a table.*

Analogous to Example 18, we choose therefore as Carr_{RM} *the full subcategory of* $\mathsf{Base}_{RM} = \mathsf{GRAPH}/M_{RM}$ *given by all* M_{RM}*-typed graphs* $(G, \tau_G : G \to M_{RM})$ *with G a subgraph of* $gr(\mathsf{Par})$*. We consider here* $gr(\mathsf{Par})$ *(and thus also G) as an element in* GRAPH_{Obj} *and not as a subgraph of* $gr(\mathsf{GRAPH})$ *(compare Remark 1). Be aware that we can have in G only maps from sets in* $\tau_G^{-1}(\boldsymbol{T})$ *to sets in* $\tau_G^{-1}(\boldsymbol{D})$ *since* $\boldsymbol{c}$ *is the only edge in* M_{RM}*!*

The semantics of a variable declaration $X \in \mathsf{Var}_{Obj}$ relative to a chosen carrier $U \in \mathsf{Carr}$ is simply the set of all **variable assignments** (keep in mind that $\mathsf{Var} \sqsubseteq \mathsf{Base}$ and $\mathsf{Carr} \sqsubseteq \mathsf{Base}$):

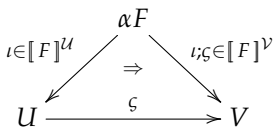
$$[\![X]\!]^U := \mathsf{Base}(X, U). \tag{1}$$

Structures for footprints are defined in full analogy to the definition of structures for signatures in traditional first-order logic.

Definition 5 (Structures)**.** *A* Ξ*-structure* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ *is given by an object U in* Carr*, the* ***carrier of*** $\mathcal{U}$*, and a family* $\Phi^{\mathcal{U}} = \{[\![F]\!]^{\mathcal{U}} \mid F \in \Phi\}$ *of sets* $[\![F]\!]^{\mathcal{U}} \subseteq \mathsf{Base}(\alpha F, U)$ *of* ***valid interpretations*** *of feature symbols F in U.*

Homomorphisms are also defined in the usual way that "truth is preserved".

Definition 6 (Homomorphisms)**.** *A* ***homomorphism*** $\varsigma : \mathcal{U} \to \mathcal{V}$ *between* Ξ*-structures is given by a morphism* $\varsigma : U \to V$ *in* Carr *such that* $\iota \in [\![F]\!]^{\mathcal{U}}$ *implies* $\iota ; \varsigma \in [\![F]\!]^{\mathcal{V}}$ *for all feature symbols F in* Φ *and all interpretations* $\iota : \alpha F \to U$*.*



Identities of carriers define identity homomorphisms and composition of homomorphisms is inherited from composition in Carr. In such a way, we obtain a category $\mathsf{Str}(\Xi)$ of all available Ξ-structures. We are, however, free to choose only those structures we are interested in (see Figure 1).

Definition 7 (Fifth parameter: Semantics)**.** *As the fifth parameter of an Institution of Statements, we choose a certain subcategory* $\mathsf{Sem}(\Xi)$ *of the category* $\mathsf{Str}(\Xi)$ *of all* Ξ*-structures.*

Example 21 (FOL: Semantics)**.** *In accordance with the traditional approach* $\mathsf{Sem}(\Xi_{FOL}) := \mathsf{Str}(\Xi_{FOL})$ *comprises all* Ξ_{FOL}*-structures* $\mathcal{U}$*, given by an arbitrary S-set U, where* $S = \{prs, nat\}$ *as in Example 11, together with arbitrary subsets* $[\![\texttt{parent}]\!]^{\mathcal{U}} \subseteq \mathsf{Set}^S((\{x_1, x_2, x_3\}, \emptyset), U)$*,* $[\![\texttt{male}]\!]^{\mathcal{U}} \subseteq \mathsf{Set}^S((\{x\}, \emptyset), U)$*,* $[\![\texttt{age}]\!]^{\mathcal{U}} \subseteq \mathsf{Set}^S((\{x_1\}, \{x_2\}), U)$ *and* $[\![\texttt{less}]\!]^{\mathcal{U}} \subseteq \mathsf{Set}^S((\emptyset, \{x_1, x_2\}), U)$*, as well as all homomorphisms between those* Ξ_{FOL}*-structures.*

Example 22 (ALC: Semantics). *Any* terminological interpretation $\mathcal{I}$ *in ALC includes the choice of a non-empty set* $\Delta^{\mathcal{I}}$*, called* domain*, an interpretation of each* concept name *in* N_C *as a subset of* $\Delta^{\mathcal{I}} \cong \mathsf{Set}(\{x\}, \Delta^{\mathcal{I}})$ *and an interpretation of each* role name *in* N_R *as a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \cong \mathsf{Set}(\{x_1, x_2\}, \Delta^{\mathcal{I}})$*. Obviously, there is a one-to-one correspondence between a terminological interpretation and a* Ξ_{FOL}*-structure in the sense of Definition 5. Homomorphisms are not considered in ALC; thus,* $\mathsf{Sem}(\Xi_{ALC}) := \mathsf{Str}(\Xi_{ALC})$ *is a discrete category.*

Example 23 (mFOL: Semantics). *In contrast to the Examples 21 and 22, we transform any carrier* (U, τ_U) *in* Carr_{mF} *(see Example 18) into exactly one corresponding* Ξ_{mF}*-structure* $\mathcal{U}$ *with:*

$$[\![\,\mathtt{un}\,]\!]^{\mathcal{U}} := \{\iota : (X_{\mathtt{un}}, \tau_{X_{\mathtt{un}}}) \to (U, \tau_U) \mid \iota(xp) \subseteq \iota(xs)\,\}$$

$$[\![\,\mathtt{bin}\,]\!]^{\mathcal{U}} := \{\iota : (X_{\mathtt{bin}}, \tau_{X_{\mathtt{bin}}}) \to (U, \tau_U) \mid \iota(xp) \subseteq \iota(xs_1) \otimes \iota(xs_2)\,\}$$

$$[\![\,\mathtt{trt}\,]\!]^{\mathcal{U}} := \{\iota : (X_{\mathtt{trt}}, \tau_{X_{\mathtt{trt}}}) \to (U, \tau_U) \mid \iota(xp) \subseteq \iota(xs_1) \otimes \iota(xs_2) \otimes \iota(xs_3)\,\}$$

$\mathsf{Str}(\Xi_{mF})$ *is given by all these* Ξ_{mF}*-structures and all homomorphims between them according to Definition 6. Note that the homomorphisms in* $\mathsf{Str}(\Xi_{mF})$ *resemble the idea of functors that preserve finite products and monomorphisms (inclusions).*

To cover the traditional approach that a predicate in a first-order structure can be an arbitrary subset of a corresponding Cartesian product of sorts (compare Example 21), we choose as $\mathsf{Sem}(\Xi_{mF})$ *the full subcategory of* $\mathsf{Str}(\Xi_{mF})$ *given by all* Ξ_{mF}*-structures* $\mathcal{U} = ((U, \tau_U), \Phi^{\mathcal{U}}_{mF})$ *such that* $\tau_U^{-1}(\boldsymbol{P})$ *is the union of all power sets* $\wp(A)$*,* $\wp(A \otimes B)$*,* $\wp(A \otimes B \otimes C)$ *with* A*,* B*,* C *ranging over all the sets in* $\tau_U^{-1}(\boldsymbol{S})$*.*

Example 24 (Category Theory: Semantics). *Analogously to Example 21,* $\mathsf{Sem}(\Xi_{CT}) := \mathsf{Str}(\Xi_{CT})$ *comprises all* Ξ_{CT}*-structures* $\mathcal{U} = (U, \Phi^{\mathcal{U}}_{CT})$ *given by an arbitrary small graph* U *together with arbitrary subsets* $[\![\,\mathtt{id}\,]\!]^{\mathcal{U}} \subseteq \mathsf{Graph}(\alpha_{CT}(\mathtt{id}), U)$*,* $[\![\,\mathtt{cmp}\,]\!]^{\mathcal{U}} \subseteq \mathsf{Graph}(\alpha_{CT}(\mathtt{cmp}), U)$*,* $[\![\,\mathtt{mon}\,]\!]^{\mathcal{U}} \subseteq \mathsf{Graph}(\alpha_{CT}(\mathtt{mon}), U)$ *and* $[\![\,\mathtt{fnl}\,]\!]^{\mathcal{U}} \subseteq \mathsf{Graph}(\alpha_{CT}(\mathtt{fnl}), U)$*. That is, we also include structures like categories without identities, categories with partial composition and so on. Moreover,* $\mathsf{Sem}(\Xi_{CT})$ *includes all homomorphisms between those* Ξ_{CT}*-structures.*

Example 25 (RM: Semantics). *Analogous to Example 23, we transform any carrier* (U, τ_U) *in* Carr_{RM} *into exactly one corresponding* Ξ_{RM}*-structure* $\mathcal{U}$*. We take, however, into account that tables do have only finitely many rows:*

$[\![\,\mathtt{tb(n)}\,]\!]^{\mathcal{U}}$ *is the set of all* M_{RM}*-typed graph homomorphisms* $\iota : \alpha_{RM}(\mathtt{tb(n)}) \to (U, \tau_U)$ *such that* $\iota(xt)$ *is a finite (!) subset of* $\bigotimes^p(\iota(xd_i) \mid 1 \leq i \leq n)$ *and the partial maps* $\iota(xc_i) : \iota(xt) \multimap \iota(xd_i)$ *are exactly the corresponding restricted projections.*

Reflecting the usual definition of foreign keys, we define $[\![\,\mathtt{fk}\,]\!]^{\mathcal{U}}$ *as the set of all* M_{RM}*-typed graph homomorphisms* $\iota : \alpha_{RM}(\mathtt{fk}) \to (U, \tau_U)$ *such that* $\iota(xc_1)(\iota(xt_1)) \subseteq \iota(xc_2)(\iota(xt_2))$*, i.e., each value in row* xc_1 *in table* xt_1 *has to appear in row* xc_2 *in table* xt_2*.*

$[\![\,\mathtt{tot}\,]\!]^{\mathcal{U}}$ *is the set of all* $\iota : \alpha_{RM}(\mathtt{tot}) \to (U, \tau_U)$ *such that* $\iota(xc) : \iota(xt) \multimap\!\!\to \iota(xd)$ *is total.* $[\![\,\mathtt{inj}\,]\!]^{\mathcal{U}}$ *comprises, correspondingly, all cases where* $\iota(xc)$ *is injective and* $[\![\,\mathtt{pk}\,]\!]^{\mathcal{U}}$ *all cases where* $\iota(xc)$ *is as well total as injective.*

As $\mathsf{Sem}(\Xi_{RM})$*, we can choose the full subcategory of* $\mathsf{Str}(\Xi_{RM})$ *given by all* Ξ_{RM}*-structures* $\mathcal{U} = ((U, \tau_U), \Phi^{\mathcal{U}}_{RM})$ *such that* $\tau_U^{-1}(\boldsymbol{T})$ *is the union of all power sets* $\wp_{fin}(\bigotimes^p(A_i \mid 1 \leq i \leq n)$ *with* $1 \leq n$ *and the* A_i*'s ranging over all the sets in* $\tau_U^{-1}(\boldsymbol{D})$*. We could require, in addition, that the sets in* $\tau_U^{-1}(\boldsymbol{D})$ *are restricted to those data types that appear in a certain version of SQL, for example.*

3.4. First-Order Feature Expressions

3.4.1. Syntax of Feature Expressions

By a feature expression, we mean something like a "formula with free variables" in traditional FOL. However, we do not consider them as formulas, but rather as derived anonymous features. For us, a formula is, semantically seen, the subject of being "valid or

not valid" in a given structure, while the semantics of a feature expression, with respect to a given structure, is the set of all its solutions, i.e., the set of all valid interpretations of the derived feature in this structure. We experience this perspective as the most adequate one when formalizing and working with constraints in Model Driven Software Engineering. Sets of solutions have also been utilized to define the validity of conditional existence equations in [7,9], for example.

Definition 8 (Feature expressions: Syntax)**.** *For a footprint $\Xi = (\Phi, \alpha)$ over* Var *we define inductively and in parallel a family* $\mathtt{FE}(\Xi)$ *of sets* $\mathtt{FE}(\Xi, X)$ *of* ***(first-order) feature Ξ-expressions*** *Ex* ***on*** *X, $X \triangleright Ex$ in symbols, where X varies over all the objects in* Var*:*

1. *Atomic expressions: $X \triangleright F(\beta)$ for any $F \in \Phi$ and any morphism $\beta : \alpha F \to X$ in* Var*.*
2. *Everything: $X \triangleright \top$ for any object X in* Var*.*
3. *Void: $X \triangleright \bot$ for any object X in* Var*.*
4. *Conjunction: $X \triangleright (Ex_1 \wedge Ex_2)$ for any expressions $X \triangleright Ex_1$ and $X \triangleright Ex_2$.*
5. *Disjunction: $X \triangleright (Ex_1 \vee Ex_2)$ for any expressions $X \triangleright Ex_1$ and $X \triangleright Ex_2$.*
6. *Implication: $X \triangleright (Ex_1 \to Ex_2)$ for any expressions $X \triangleright Ex_1$ and $X \triangleright Ex_2$.*
7. *Negation: $X \triangleright \neg Ex$ for any expression $X \triangleright Ex$.*
8. *Quantification: $X \triangleright \exists(\varphi, Y : Ex)$ and $X \triangleright \forall(\varphi, Y : Ex)$ for any expression $Y \triangleright Ex$ and any morphism $\varphi : X \to Y$ in* Var *that is not an isomorphism.*

Remark 6 (Notation for expressions)**.** *In traditional FOL, X and Y are sets of variables and, instead of arbitrary maps $\varphi : X \to Y$, only inclusion maps $\varphi = in_{X,Y} : X \hookrightarrow Y$ are considered. Moreover, only the quantified variables $Y \setminus X$ are recorded while Y has to be (re)constructed as the union $X \cup (Y \setminus X)$. In other words, our Y lists all (!) variables that are allowed to appear as* free variables *in Ex! We record the whole Y for three reasons: (1) Already in* Graph *(not to talk about arbitrary presheaf topoi), we do not have complements; (2) We quantify actually over morphisms with source Y when we define the semantics of quantifications (compare Definition 10); (3) In contrast to traditional FOL, $\varphi : X \to Y$ is allowed to be non-monic.*

We allow non-monic morphims $\varphi : X \to Y$ to express identifications. In such a way, we can survive, for the moment, without explicit equations even in cases where Var *is a subcategory of a set-based category. We illustrate this mechanism in the Examples 26 and 29.*

If Var *is a subcategory of a set-based category, like* Set*,* $[S \to$ Set$]$*,* Set$/M_{mF}$*,* Graph *or* Graph$/M_{RM}$*, for example, variable declarations X are constituted by single entities; thus, we can talk about individual "variables". Moreover, inclusions of sets give us corresponding* inclusion morphisms *at hand. In case, $\varphi = in_{X,Y} : X \hookrightarrow Y$ is such an inclusion morphism we will drop φ (see Examples 26, 28 and 29).*

If $\varphi : X \to Y$ is an isomorphism, quantification is obsolete; thus, we excluded those cases.

Remark 7 (Everything and Void)**.** *For the definition of sketch conditions in Section 6, we need another pair of symbols for "true" and "false"; thus, we decided to use for feature expressions the symbols $\top$ and $\bot$, respectively.*

We consider $\top$ and $\bot$ not as logical constants *but as* special feature symbols*, inbuilt in any Institution of Statements (analogously to the equation symbol in Universal Algebra).*

To make this statement fully precise, we have to assume that Base*, and thus also* Var*, has an initial object* **0***. **0** is then the arity of $\top$ and $\bot$, while the fixed semantics for any carrier U is given by the two subsets of the singleton* $\mathsf{Base}(\mathbf{0}, U) = \{!_U : \mathbf{0} \to U\}$*, namely $[\![\bot]\!]^U = \emptyset$ and $[\![\top]\!]^U = \{!_U\}$. Consequently, we could use then the same notation as for atomic expressions, namely $X \triangleright \top(!_X)$ and $X \triangleright \bot(!_X)$ where $!_X : \mathbf{0} \to X$ is the unique initial morphism into X.*

Remark 8 (Closed expressions: Syntax)**.** *If* Base *has an initial object* **0***, feature expressions of the form $\mathbf{0} \triangleright Ex$ will be called* ***closed expressions****. Note that quantification will generate closed expressions only in case $X = \mathbf{0}$ where $\varphi = !_Y : \mathbf{0} \to Y$ is the only choice for φ, in this case.*

Example 26 (FOL: Expressions)**.** *We intend to illustrate that and how traditional first-order formulas appear in our framework. First, we consider only those cases where the morphism φ in quantifications is an inclusion morphisms and will be therefore dropped.*

In Example 11, we proposed to represent finite S-sorted sets by lists of variable declarations. The arity $\alpha_{FOL}(\texttt{parent}) := (\{x_1, x_2, x_3\}, \emptyset)$ for the feature symbol $\texttt{parent} \in \Phi_{FOL}$, for example, is represented by $[x_1\text{: } prs, x_2\text{: } prs, x_3\text{: } prs]$. Pursuing the idea to consider a tuple as a convenient notation for an "associative array", we can denote the atomic expression $\texttt{parent}(\beta)$, with $\beta : \alpha_{FOL}(\texttt{parent}) \to Y$ an $\{prs, nat\}$-map, simply as $\texttt{parent}(\beta(x_1), \beta(x_2), \beta(x_3))$.

Relying on this notational convention, we obtain, for example, the closed Ξ_{FOL}-expression $\mathbf{0} \triangleright \forall([x_1 : prs, x_2 : prs, x_3 : prs, y_1 : nat, y_2 : nat] :$
$$((\texttt{parent}(x_1, x_2, x_3) \wedge \texttt{age}(x_1, y_1)) \wedge \texttt{age}(x_2, y_2)) \longrightarrow \texttt{less}(y_1, y_2))$$
(with $\mathbf{0}$ the empty S-set) expressing that a child is always younger than a parent.

Our main point, however, is to consider feature expressions as derived features *enabling us to denote properties in an anonymous way. The following feature Ξ_{FOL}-expression younger, for example, gives us the property* younger than *at hand by hiding the exact age of a person:*

$$[y : prs, x : nat] \triangleright \exists([y : prs, x : nat, x_1 : nat] : (\texttt{less}(x, x_1) \wedge \texttt{age}(y, x_1)))$$

The next feature Ξ_{FOL}-expression sbl provides the property being a sibling of someone*:*

$$[y : prs] \triangleright \exists([y : prs, x_1 : prs, x_2 : prs, x_3 : prs] : \texttt{parent}(y, x_2, x_3) \wedge \texttt{parent}(x_1, x_2, x_3) \wedge \neg\exists(\varphi, [x : prs, x_2 : prs, x_3 : prs] : \top))$$

with $\varphi : [y : prs, x_1 : prs, x_2 : prs, x_3 : prs] \to [x : prs, x_2 : prs, x_3 : prs]$ defined by the assignments $y, x_1 \mapsto x; x_2 \mapsto x_2; x_3 \mapsto x_3$. Note that the Ξ_{FOL}-expression $\neg\exists(\varphi, [x : prs, x_2 : prs, x_3 : prs] : \top)$ on $[y : prs, x_1 : prs, x_2 : prs, x_3 : prs]$ encodes the inequality $\neg(y = x_1)$.

For convenience, we could introduce an auxiliary feature symbol `sibling` *with arity $[y : prs]$ and use $\texttt{sibling}(y)$ as a shorthand (macro) for this derived feature expression. The conjunction $(\texttt{male}(y) \wedge \texttt{sibling}(y))$ would then represent a unary property* being brother of someone*. To ensure that then any feature expression $X \triangleright Ex$, containing the auxiliary feature symbol* `sibling`*, can be expanded into an equivalent feature expression $X \triangleright Ex'$, containing only the original feature symbols* `male` *and* `parent`*, we need a corresponding* substitution mechanism*.*

Remark 9 (Substitution)**.** *Fortunately, we do not need substitution mechanisms to define Institutions of Statements and to utilize them for specifications purposes. We need, essentially, only a category as we show and demonstrate it in this paper. To develop, however, fully fledged and practical Logics of Statements and, especially, corresponding deduction calculi, we will need appropriate substitution mechanisms.*

An exhaustive and systematic study on what additional categorical infrastructure we have to presuppose to have handy substitution mechanisms at hand is out of range for this paper. In Appendix A, we present, nevertheless, some first observations, definitions and constructions.

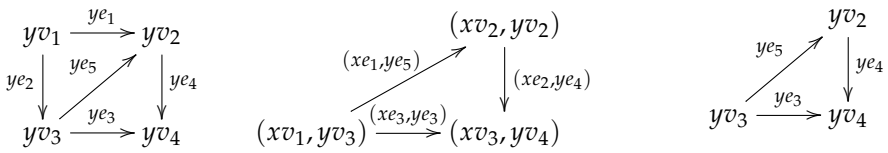
Example 27 (ALC: Expressions)**.** *ALC focuses on* derived concepts*, i.e., in our view, on feature expressions with X a singleton. To describe, however, all derived concepts as feature expressions, we have to use arbitrary finite sets of variables and inclusions between them. We outline the standard encoding of ALC in FOL. Using our notational conventions, the ALC construct "universal restriction $\forall\texttt{R}.\texttt{C}$ for any role $\texttt{R} \in N_R$, any (derived) concept $\texttt{C}$", can be described as follows: For any role $\texttt{R}$ in N_R, any expression $\{y\} \triangleright \texttt{C}$ and any variables x_1, x_2, not appearing in $\texttt{C}$, we have: $\{x_1\} \triangleright \forall(\{x_1, x_2\} : \texttt{R}(x_1, x_2) \to \texttt{C}_\psi(x_2))$ where $\psi : \{y\} \to \{x_1, x_2\}$ is given by $\psi(y) = x_2$ and the expression $\{x_1, x_2\} \triangleright \texttt{C}_\psi(x_2)$ is obtained by substituting each occurrence of y in $\texttt{C}$ by x_2 and by extending each variable declaration Y in $\texttt{C}$ by the "fresh variable" x_1 (compare Appendix A). Analogously, the ALC construct "the existential restriction $\exists\texttt{R}.\texttt{C}$ of a concept $\texttt{C}$ by a role $\texttt{R} \in N_R$" can be described by existential quantification: For any role $\texttt{R}$ in N_R, any expression $\{y\} \triangleright \texttt{C}$ and any variables x_1, x_2, not appearing in $\texttt{C}$, we have: $\{x_1\} \triangleright \exists(\{x_1, x_2\} : \texttt{R}(x_1, x_2) \wedge \texttt{C}_\psi(x_2))$.*

Example 28 (mFOL: Expressions)**.** *This is an example where we do not need the full first-order power. Actually, we only need atomic feature Ξ_{mF}-expressions to state that a set is the subset of a unary, binary or tertiary product of other sets.*

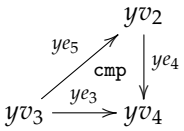
Example 29 (CT: Expressions)**.** *To support the shift of paradigm from string-based to diagrammatic logic was one of our main motivations to develop our framework. Therefore, we will spend a bit more space and put some more effort on this example.*

Representation and visualization of graph homomorphisms: For a finite graph A, we can represent and visualize a graph homomorphism $\varphi : A \to B$ by means of the corresponding graph of assignments *$A^\varphi = (A^\varphi_V, A^\varphi_E, sc^{A^\varphi}, tg^{A^\varphi})$ with $A^\varphi_V := \{(v, \varphi_V(v)) \mid v \in A_V\}$, $A^\varphi_E := \{(e, \varphi_E(e)) \mid e \in A_E\}$ where sc^{A^φ} and tg^{A^φ} are defined for all $e \in A_E$ by $sc^{A^\varphi}(e, \varphi_E(e)) = (sc^A(e), \varphi_V(sc^A(e)))$ and $tg^{A^\varphi}(e, \varphi_E(e)) = (tg^A(e), \varphi_V(tg^A(e)))$, respectively. The graphs A and A^φ are isomorphic by construction. Note that we actually simply lift the idea of "tuples as associative arrays" to graphs instead of sets of indexes.*

We consider the graph Y, visualized below on the left. For the graph morphism $\varphi : \alpha(\texttt{cmp}) \to Y$, defined by the assignments $xv_1 \mapsto yv_3$, $xv_2 \mapsto yv_2$, $xv_3 \mapsto yv_4$, $xe_1 \mapsto ye_5$, $xe_2 \mapsto ye_4$, $xe_3 \mapsto ye_3$, the corresponding graph of assignments $\alpha(\texttt{cmp})^\varphi$ is visualized below in the middle. In many cases, we can fortunately use for $\alpha(\texttt{cmp})^\varphi$ the shorthand graph*, on the right, without causing unambiguities.*

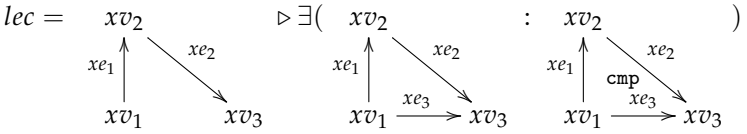


As proposed in [3], we can also work with a sequential representation of the shorthand graph (compare also Example 26): We can represent finite graphs by a list of edges plus a list of vertexes, respectively. Pursuing the idea of tuples as associative arrays*, a graph homomorphism $\varphi : A \to B$ is then denoted by a list of image edges and a list of image vertexes in graph B.*

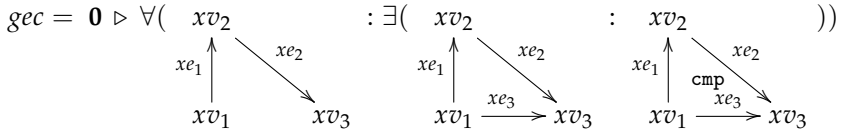


In such a way, we can visualize the atomic Ξ_{CT}-expression $\texttt{cmp}(\varphi)$ by the graph above and represent it also by the string $\texttt{cmp}(ye_5, ye_4, ye_3; yv_3, yv_2, yv_4)$. Since $\alpha(\texttt{cmp})$ has no isolated vertexes, $\varphi : \alpha(\texttt{cmp}) \to Y$ is uniquely determined by the edge-assignments; thus, we could even use the shorthand notation $\texttt{cmp}(ye_5, ye_4, ye_3)$ instead.

Ξ_{CT}-Expressions: The local *property* composition is defined for a certain pair of edges *can be formalized by the following feature Ξ_{CT}-expression:*

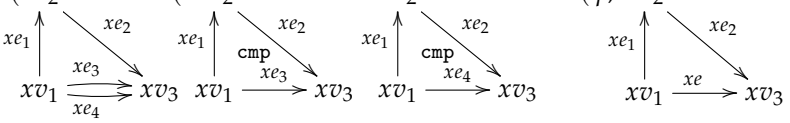


Universal quantification transforms this property into a general *property* composition is always defined *formalized by the following feature Ξ_{CT}-expression, where* **0** *is the empty graph:*



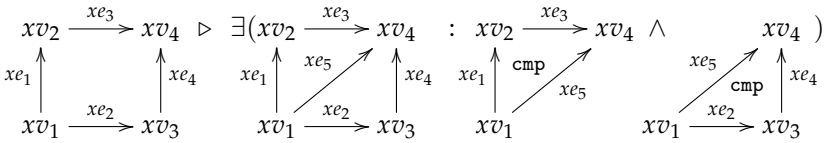
The general property composition is always unique *is given by the expression guc:*

$$guc = \mathbf{0} \triangleright \forall(xv_2 : (xv_2 \wedge xv_2 \rightarrow \exists(\varphi, xv_2 : \top)))$$



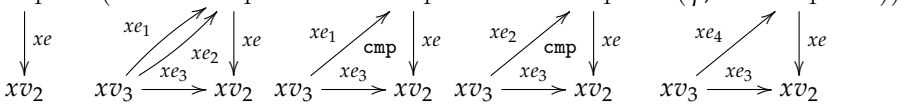
where φ simply maps xe_3 and xe_4 to xe. Analogously, we can also represent the other axioms of categories – existence and uniqueness of identity morphisms, both identity laws and the associativity law – by means of feature Ξ_{CT}-expressions. In addition, feature expressions are a handy tool to hide auxiliary items *in diagrammatic specifications. The property* commutative square*, for example, is given by the feature Ξ_{CT}-expression csq, where we hide the diagonal:*

$$csq = xv_2 \xrightarrow{xe_3} xv_4 \triangleright \exists(xv_2 \xrightarrow{xe_3} xv_4 : xv_2 \xrightarrow{xe_3} xv_4 \wedge xv_4)$$



That concepts and constructions are defined by ***universal properties*** *is the crucial characteristic of Category Theory as a modeling technique. The concept* monomorphism*, for example, is defined by the feature Ξ_{CT}-expression mon:*

$$mon = xv_1 \triangleright \forall(xv_1 : xv_1 \wedge xv_1 \rightarrow \exists(\varphi, xv_1 : \top))$$



where φ maps xe_1 and xe_2 to xe_4. In most cases, however, a universal property is the conjunction of a universally quantified existence assertion and a universally quantified uniqueness assertion (see Remark 10). The concept final object*, for example, is defined by the feature Ξ_{CT}-expression fnl where φ maps xe_1 and xe_2 to xe:*

$$fnl = xv \triangleright \forall(xv_1\ xv : \exists(xv_1 \xrightarrow{xe} xv : \top)) \wedge \forall(xv_1 \underset{xe_2}{\overset{xe_1}{\rightrightarrows}} xv : \exists(\varphi, xv_1 \xrightarrow{xe} xv : \top))$$

In case, we want to work with an explicit property two parallel morphisms are equal*, we are free to utilize the Ξ_{CT}-expression $[=]$ where φ maps xe_1 and xe_2 to xe:*

$$[=] = xv_1 \underset{xe_2}{\overset{xe_1}{\rightrightarrows}} xv_2 \quad \triangleright \quad \exists(\varphi, xv_1 \xrightarrow{xe} xv_2 : \top)$$

Remark 10 (Limits and Colimits)**.** *The fact that the universal properties in Category Theory do have a uniform and relatively simple logical structure shaped the theory of generalized sketches in [15]. The main ingredients of the definition of (co)limits are categorical diagrams, i.e., graph homomorphisms, thus we can beneficially use feature Ξ_{CT}-expressions to characterize the logical structure of the concept (co)limit.*

The universal property, defining a finite (co)limit, is a conjunction of two assertions – existence of mediators and uniqueness of mediators. We can express those assertions by feature Ξ_{CT}-expressions with the following structure (compare the definition of the concept final object *in Example 29):*

$$exist_I := C_I \triangleright \forall(C_I + C_I' : Ex_1 \longrightarrow \exists(C_I \overset{\rightarrow}{+} C_I' : Ex_2))$$
$$unique_I := C_I \triangleright \forall(C_I \overset{\Rightarrow}{+} C_I' : Ex_3 \longrightarrow \exists(\varphi, C_I \overset{\rightarrow}{+} C_I' : \top))$$

I is the shape graph of the (co)limit, i.e., the empty graph in the case of final objects. C_I adds to I the shape of a (co)cone with base I while $C_I + C_I'$ extends C_I with the shape of a second (co)cone with base I. Ex_1 is the conjunction of all atomic cmp*-expressions on $C_I + C_I'$ turning both (co)cones into commutative ones. $C_I \overset{\rightarrow}{+} C_I'$ extends $C_I + C_I'$ by a single mediator while Ex_2 is the conjunction of all atomic* cmp*-expressions on $C_I \overset{\rightarrow}{+} C_I'$ expressing the commutativity requirements for the mediator. $C_I \overset{\Rightarrow}{+} C_I'$ extends $C_I + C_I'$ by two parallel mediators and Ex_3 is the conjunction of all*

atomic cmp*-expressions on* $C_I \overset{\Rightarrow}{+} C'_I$ *expressing the commutativity requirements for both mediators.* $\varphi : C_I \overset{\Rightarrow}{+} C'_I \longrightarrow C_I \overset{\rightarrow}{+} C'_I$ *simply identifies the two mediators in* $C_I \overset{\Rightarrow}{+} C'_I$.

Example 30 (RM: Expressions)**.** *To formalize declarations of tables and data base schemes, respectively, we need only atomic feature* Ξ_{RM}*-expressions; thus, we consider in this example only atomic* Ξ_{RM}*-expressions. To deal also with so-called* business rules*, we would need, however, the full spectrum of first-order* Ξ_{RM}*-expressions.*

As seen in the examples, there are cases where we need only a restricted selection of first-order feature expressions. The freedom to select only the feature expressions we are interested in establishes a new parameter (see Figure 1).

Definition 9 (Sixth parameter: Choice of expressions)**.** *As the sixth parameter of an Institution of Statements, we choose an* Var_{Obj}*-indexed family* $\mathtt{XE}(\Xi)$ *of subsets* $\mathtt{XE}(\Xi, X) \subseteq \mathtt{FE}(\Xi, X)$ *of first-order* Ξ*-expressions on* $X \in \mathsf{Var}_{Obj}$.

Despite the fact that the family $\mathtt{FE}(\Xi, X)$, $X \in \mathsf{Var}_{Obj}$ of sets of first-order feature expressions is defined by mutual induction, there is no explicit relationship between the different sets $\mathtt{FE}(\Xi, X)$ since we do not base the definition of our framework on translation maps induced by variables translations, i.e., morphisms in Var (see Definition A1 in Appendix A). Therefore, the choice of $\mathtt{XE}(\Xi, X)$ for a certain X can be made independently from all the other choices! However, if we also incorporate later translation maps, it will be reasonable to require that the choices of the different $\mathtt{XE}(\Xi, X)$ are compatible with translation maps!

What are natural choices? We could simply choose all first-order feature expressions, i.e., $\mathtt{XE}(\Xi, X) = \mathtt{FE}(\Xi, X)$ for all $X \in \mathsf{Var}_{Obj}$, as we will do it in the FOL-example as well as in the CT-example. The other extreme case is to forget about "first-order" and to restrict ourselves to atomic feature expressions. This we have done in [13] and in DPF [18,21] since first-order feature expressions have not been available. For the mFOL-example and the RM-example, it is sufficient to use atomic expressions only.

Besides these two extreme cases, we could, for example, exclude negation or we could choose a minimal set of logical connectives and so on. In the ALC-example, we choose only the first-order feature expressions necessary to encode ALC in first-order logic (compare Example 27).

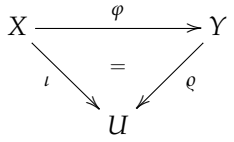
If Base has an initial object, we could restrict ourselves to closed expressions only (see Remark 8). In this case, we are back to traditional institutions since we do not need contexts to utilize closed formulas for specification purposes. The definition of closed formulas and of the satisfaction relation for closed formulas goes, however, always via open formulas and therefore any deduction calculus for closed formulas is based on a manipulation of open formulas. In other words: We are convinced that the concept of a context, defined in Definition 11, is relevant and beneficial for any logic beyond propositional logic even for traditional first-order predicate logic!

3.4.2. Semantics of Feature Expressions

Due to Definition 5, a Ξ-structure $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ fixes for each feature symbol $F \in \Phi$ its semantics in $\mathcal{U}$ as a set $[\![F]\!]^{\mathcal{U}} \subseteq \mathsf{Base}(\alpha F, U)$ of all valid interpretations of F in U.

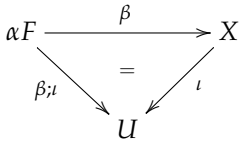
Relying on the inductive definition of first-order feature expressions, we can extend the semantics of feature symbols and define the semantics of a feature expression $X \triangleright Ex$ in a Ξ-structure $\mathcal{U}$ as a set $[\![Ex]\!]^{\mathcal{U}}_X$ of all **valid interpretations (solutions)** of $X \triangleright Ex$ in $\mathcal{U}$. This semantics is a restriction of the semantics of X relative to the carrier U as defined by Equation (1), i.e., $[\![Ex]\!]^{\mathcal{U}}_X \subseteq [\![X]\!]^U = \mathsf{Base}(X, U)$. For **interpretations** $\iota : X \to U$, we will use, instead of $\iota \in [\![Ex]\!]^{\mathcal{U}}_X$, also the more traditional notation $\iota \models^{\mathcal{U}} X \triangleright Ex$.

Given a morphism $\varphi : X \to Y$ in Var, we say that an interpretation $\varrho : Y \to U$ is an **expansion** of an interpretation $\iota : X \to U$ **via** φ if, and only if, $\varphi ; \varrho = \iota$.

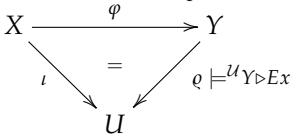


Definition 10 (Feature expressions: Semantics)**.** *The semantics of feature Ξ-expressions in an arbitrary, but fixed, Ξ-structure $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ is defined inductively:*

1. *Atomic expressions: $\iota \in [\![F(\beta)]\!]^{\mathcal{U}}_X$ iff $\beta;\iota \in [\![F]\!]^{\mathcal{U}}$*



2. *Everything: $[\![\top]\!]^{\mathcal{U}}_X := [\![X]\!]^U = \mathsf{Base}(X, U)$*
3. *Void: $[\![\bot]\!]^{\mathcal{U}}_X := \emptyset$*
4. *Conjunction: $[\![(Ex_1 \wedge Ex_2)]\!]^{\mathcal{U}}_X := [\![Ex_1]\!]^{\mathcal{U}}_X \cap [\![Ex_2]\!]^{\mathcal{U}}_X$*
5. *Disjunction: $[\![(Ex_1 \vee Ex_2)]\!]^{\mathcal{U}}_X := [\![Ex_1]\!]^{\mathcal{U}}_X \cup [\![Ex_2]\!]^{\mathcal{U}}_X$*
6. *Implication: $\iota \in [\![Ex_1 \to Ex_2]\!]^{\mathcal{U}}_X$ iff $\iota \in [\![Ex_1]\!]^{\mathcal{U}}_X$ implies $\iota \in [\![Ex_2]\!]^{\mathcal{U}}_X$*
7. *Negation: $[\![\neg Ex]\!]^{\mathcal{U}}_X := \mathsf{Base}(X, U) \setminus [\![Ex]\!]^{\mathcal{U}}_X$*
8. *Existential quantification: $\iota \in [\![\exists(\varphi, Y : Ex)]\!]^{\mathcal{U}}_X$ iff there exists an expansion ϱ of ι via φ such that $\varrho \in [\![Ex]\!]^{\mathcal{U}}_Y$.*



Universal quantification: $\iota \in [\![\forall(\varphi, Y : Ex)]\!]^{\mathcal{U}}_X$ iff for all expansions ϱ of ι via φ we have $\varrho \in [\![Ex]\!]^{\mathcal{U}}_Y$.

Remark 11 (Feature expressions: Semantics)**.** *Every feature symbol F in Φ reappears as the Ξ-expression $\alpha F \triangleright F(id_{\alpha F})$ and Definition 10 ensures $[\![F(id_{\alpha F})]\!]^{\mathcal{U}}_{\alpha F} = [\![F]\!]^{\mathcal{U}}$.*

The universal quantification $X \triangleright \forall(\varphi, Y : Ex)$ is trivially valid if there is no expansion of ι via φ at all, while the existential quantification $X \triangleright \exists(\varphi, Y : Ex)$ is not valid, in this case.

*Two expressions $X \triangleright Ex_1$ and $X \triangleright Ex_2$ are **semantical equivalent**, $X \triangleright Ex_1 \equiv Ex_2$ in symbols, if, and only if, $[\![Ex_2]\!]^{\mathcal{U}}_X = [\![Ex_2]\!]^{\mathcal{U}}_X$ for all Ξ-structures $\mathcal{U}$ in $\mathsf{Sem}(\Xi)$. Definition 10 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 already at some places in the examples.*

Remark 12 (Closed expressions: Semantics)**.** *For a closed expression $\mathbf{0} \triangleright Ex$ (see Remark 8), $[\![\mathbf{0}]\!]^U = \mathsf{Base}(\mathbf{0}, U)$ is a singleton with the initial morphism $!_U : \mathbf{0} \to U$ as the only element. In such a way, we have either $[\![Ex]\!]^{\mathcal{U}}_{\mathbf{0}} = [\![\top]\!]^{\mathcal{U}}_{\mathbf{0}} = \{!_U\}$, i.e., $!_U \models^{\mathcal{U}} \mathbf{0} \triangleright Ex$, or $[\![Ex]\!]^{\mathcal{U}}_{\mathbf{0}} = [\![\bot]\!]^{\mathcal{U}}_{\mathbf{0}} = \emptyset$, i.e., $!_U \not\models^{\mathcal{U}} \mathbf{0} \triangleright Ex$.*

3.5. Institutions of Statements

Generalizing concepts like *set of generators* in Group Theory, *underlying graph of a sketch* in Category Theory, *set of individual names* in Description Logics and *underlying graph of a model* in Software Engineering, we introduce in this section the concept of a *context* as one of our main conceptual and methodological proposals. Furthermore, we introduce the concept *statement (in a context)* in generalizing the corresponding concepts defining relation in Group Theory, diagram in a sketch in Category Theory, concept/role assertion in Description Logic and constraint in Software Engineering. We use institutions [2,31] as a methodological guideline to define and present the formalisms build upon these new concepts.

3.5.1. Category of Contexts and Sentence Functor

As **abstract signatures** in an Institution of Statements, we introduce contexts.

Definition 11 (Seventh parameter: Contexts)**.** *As the seventh parameter of an Institution of Statements, we choose another subcategory* Cxt *of* Base*. The objects in* Cxt *are called* ***contexts*** *while we refer to the morphisms in* Cxt *as* ***context morphisms****.*

If Base *has initial objects, we assume that* Cxt *contains, at least, one of them denoted by* **0***.*

Even if Cxt is called the "seventh parameter", the choice of Cxt relies only on the chosen Base and does not depend on all the other parameters we introduced (see Figure 1)!

Remark 13 (Variables vs. context vs. carrier)**.** *Introducing contexts, we establish a technological layer independent of "pure syntax" (variables) and "pure semantics" (carriers of structures) as we postulated it in Requirement 2 (p. 3). We prefer to consider variable declarations as something finite or enumerable while contexts can be arbitrary.*

In case Var *is a subcategory of* Cxt*, we perceive the inclusion* $\mathsf{Var} \sqsubseteq \mathsf{Cxt}$ *as a* change of roles*: Variables are essentially syntactic items but can also serve as* generators of structures*, like groups and (term) algebras, for example.*

If we are interested in completeness proofs and corresponding freely generated structures, we have to suppose $\mathsf{Carr} \sqsubseteq \mathsf{Cxt}$*. Coming back to the discussion in Section 1.1.1, the introduction of contexts allows us to keep syntax and semantics separated and to avoid, in such a way, certain kinds of circularity in the definition of formalisms.*

Example 31 (FOL: Context)**.** *PROLOG distinguishes between* atomic values (literals) *and* (logical) variables*. Literals can be either* number literals *or* symbolic literals*.*

Our choice of contexts reflects this line of tradition. We define Cxt_{FOL} *as the subcategory of* $\mathsf{Carr}_{FOL} = \mathsf{Base}_{FOL} = \mathsf{Set}^S$ *given by all S-sets* $K = (K_s \mid s \in S)$ *with* K_s *a finite set of literals and logical variables for all* $s \in S$*.*

For the sample footprint $\Xi_{FOL} = (\Phi_{FOL}, \alpha_{FOL})$ *with* $S = \{prs, nat\}$ *(see Example 11), we consider a sample context* K *with* K_{nat} *the set of all natural numbers from* 0 *to* 200 *and* $K_{prs} = \{Anna, Michael, Dora, Heinz, Sorin, Gabi, Uwe\}$*.*

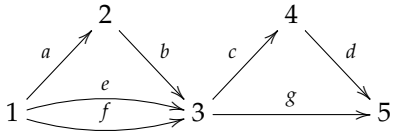
Example 32 (ALC: Context)**.** *This example has been chosen since it works explicitly with contexts in our sense. ALC uses the term* individual name *instead of* symbolic literal *and contexts in ALC are* sets N_O of individual names*.*

Example 33 (mFOL: Context)**.** *In this example, we describe the traditional formalism* many-sorted first-order logic without functions *as such; thus, a context should declare finite sets of sort and predicate symbols, respectively.*

Analogously to the definition of Var_{mF} *in Example 8, we assume an enumerable set PSym of admissible predicate symbols and an enumerable set SSym of admissible sort symbols.* Cxt_{mF} *is then the full subcategory of* Set/M_{mF} *given by all finite* M_{mF}*-typed sets* $(K, \tau_K : K \to M_{mF})$ *such that* $\tau_K^{-1}(\boldsymbol{S}) \subseteq SSym$ *and* $\tau_K^{-1}(\boldsymbol{P}) \subseteq PSym$*.*

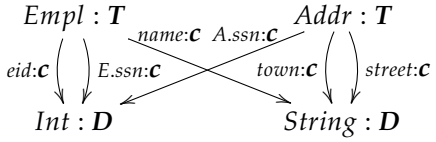
To be able to reconstruct the sample FOL-footprint $\Xi_{FOL} = (\Phi_{FOL}, \alpha_{FOL})$ *(see Example 11), we choose for the sample mFOL-footprint* $\Xi_{mF} = (\Phi_{mF}, \alpha_{mF})$ *in Example 13 the sample context* $(K, \tau_K : K \to M_{mF})$ *with* $\tau_K^{-1}(\boldsymbol{S}) := \{prs, nat\}$ *and* $\tau_K^{-1}(\boldsymbol{P}) := \{\texttt{parent}, \texttt{male}, \texttt{age}, \texttt{less}\}$*.*

Example 34 (CT: Context)**.** *We simply choose* $\mathsf{Cxt}_{CT} := \mathsf{Carr}_{CT} = \mathsf{Base}_{CT} = \mathsf{Graph}$*. As an example, we consider the following finite graph G.*



Example 35 (RM: Context)**.** *A context in this example declares the items in a database schema, i.e., a finite graph with table identifiers, datatype identifiers, and column identifiers, respectively. Analogously to Example 33, we assume an enumerable sets TId of admissible table identifiers, DId of admissible datatype identifiers and CId of admissible column identifiers, respectively.*

As Cxt_{RM}, we choose the full subcategory of Graph/M_{RM} given by all finite M_{RM}-typed graphs $(K, \tau_K : K \to M_{RM})$ such that $\tau_K^{-1}(\boldsymbol{T}) \subseteq TId$, $\tau_K^{-1}(\boldsymbol{D}) \subseteq DId$ and $\tau_K^{-1}(\boldsymbol{c}) \subseteq CId$. We intend to formalize the database schema, discussed in Section 3.1.5, and consider the sample RM-context (K, τ_K) as depicted in the following diagram.



Note that both tables do have a column with name ssn; thus, we distinguish between them by means of the table identifiers. □

Feature expressions can be utilized to make statements in a certain context. Those *statements in context* are the **sentences** in an Institution of Statements.

Definition 12 (Statement)**.** *An $\mathtt{XE}(\Xi)$-**statement** (X, Ex, γ) **in context** $K \in \mathsf{Cxt}_{Obj}$ is given by a feature Ξ-expression $X \triangleright Ex$ in $\mathtt{XE}(\Xi, X)$ and a **binding morphism** $\gamma\colon X \to K$ in* Base*.*

By $\mathtt{Stm}(K)$, we denote the set of all $\mathtt{XE}(\Xi)$-statements in K.

Statements are part of sketches and examples of sketches are presented in Section 5.

Remark 14 (Atomic statements)**.** *For a feature symbol $F \in \Phi$ and a context K there can be different variable declarations X, X', morphisms $\beta : \alpha F \to X$, $\beta' : \alpha F \to X'$ and binding morphisms $\gamma : X \to K$, $\gamma' : X' \to K$ such that $\beta;\gamma = \beta';\gamma'$. That is, the distinct statement expressions $(X, F(\beta), \gamma)$ and $(X', F(\beta'), \gamma')$ represent somehow the "same statement" in K.*

*We choose therefore a kind of **normal form** to define the concept* atomic statement*: **Atomic statements** in context K are statements of the form $(\alpha F, F(id_{\alpha F}), \gamma)$, $\gamma : \alpha F \to K$. For any context K we denote by $\mathtt{At}(K)$ the set of all atomic statements in K.*

In abuse of notation, we will sometimes use for atomic statements the same notation $F(\gamma)$ as for atomic expressions. Thus, we can, in the examples, take advantage of our notational conventions based on the idea of "associative arrays".

Remark 15 (General statements and closed formulas)**.** *If* Base *has an initial object $\mathbf{0}$, there is for any closed expression $\mathbf{0} \triangleright Ex$ (see Remarks 8 and 12) a unique initial morphism $\gamma = !_K : \mathbf{0} \to K$; thus, we have $(\mathbf{0}, Ex, !_K) \in \mathtt{Stm}(K)$ for any context K and all the closed expressions $\mathbf{0} \triangleright Ex$ in $\mathtt{XE}(\Xi, \mathbf{0})$. We call $(\mathbf{0}, Ex, !_K)$ a **general statement in** K.*

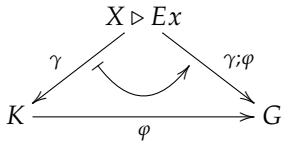
*The general statements in $\mathtt{Stm}(\mathbf{0})$, i.e., statements of the form $(\mathbf{0}, Ex, id_{\mathbf{0}})$ are the precise formal counterpart of traditional **closed formulas** within our framework. Be aware that there may be statements $(X, Ex, \gamma : X \to \mathbf{0})$ in $\mathtt{Stm}(\mathbf{0})$ with X non-initial.*

Remark 16 (Expression vs. statement)**.** *The idea behind our definition of statements is to encapsulate the relatively intricate construction of first-order syntactic entities and do it once and for all. In such a way, we achieve the following objectives: (1) There is no need to lift arbitrary "semantic entities", like elements in the carrier of a structure, to the syntactic level. (2) We can define and work with first-order statements in arbitrary base categories. (3) We do not depend on translation maps (compare Definition A1 in Appendix A) to translate first-order statements. (4) The translation of first-order statements is simply performed by composition in the category* Base*!*

This encapsulation trick we have seen in [31] where it is used for "initial/free constraints".

Any morphism $\varphi\colon K \to G$ in Cxt induces a map $\mathtt{Stm}(\varphi)\colon \mathtt{Stm}(K) \to \mathtt{Stm}(G)$ defined by simple post-composition for all statements (X, Ex, γ) in K:

$$\mathtt{Stm}(\varphi)(X, Ex, \gamma) := (X, Ex, \gamma; \varphi) \tag{2}$$



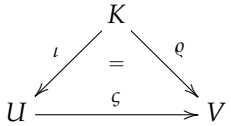
It is easy to show that the assignments $K \mapsto \mathtt{Stm}(K)$ and $\varphi \mapsto \mathtt{Stm}(\varphi)$ provide a functor $\mathtt{Stm}\colon \mathsf{Cxt} \to \mathsf{Set}$. This is the **sentence functor** of an Institution of Statements.

3.5.2. Model Functor

Interpretations of contexts are the **models** in an Institution of Statements.

Definition 13 (Context interpretations)**.** *An **interpretation** $(\iota, \mathcal{U})$ **of a context** $K \in \mathsf{Cxt}_{Obj}$ is given by a Ξ-structure $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ in $\mathsf{Sem}(\Xi)$ and a morphism $\iota\colon K \to U$ in Base.*

A morphism $\varsigma\colon (\iota, \mathcal{U}) \to (\varrho, \mathcal{V})$ between two interpretations of K is given by a morphism $\varsigma\colon \mathcal{U} \to \mathcal{V}$ in $\mathsf{Sem}(\Xi)$ such that $\iota; \varsigma = \varrho$ for the underlying morphism $\varsigma\colon U \to V$ in Carr.



For any context K in Cxt we denote by $\mathtt{Int}(K)$ the category of all interpretations of K and all morphisms between them and by $\Pi_K : \mathtt{Int}(K) \to \mathsf{Sem}(\Xi)$ the obvious projection functor.

Note that, for an initial object $K = \mathbf{0}$, the projection functor $\Pi_{\mathbf{0}} : \mathtt{Int}(\mathbf{0}) \to \mathsf{Sem}(\Xi)$ is an isomorphism.

For any Ξ-structure $\mathcal{U}$ in $\mathsf{Sem}(\Xi)$, the corresponding **fiber over** $\mathcal{U}$, i.e., the subcategory of $\mathtt{Int}(K)$ given by all interpretations of K in $\mathcal{U}$, is a discrete category representing the hom-set $\mathsf{Base}(K, U)$.

Remark 17 (Functorial semantics)**.** *We present in this paper an abstract and general definition of Institutions of Statements covering a brought range of applications. Therefore, we are not assuming any structure on the hom-sets $\mathsf{Base}(K, U)$.*

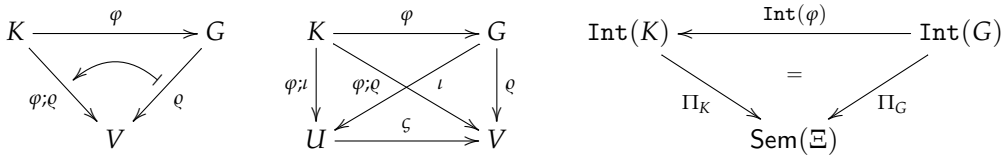
In examples, following the path of Functorial Semantics, $\mathsf{Sem}(\Xi)$ will be constituted by Ξ-structures $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ where U is provided by a category like Set or Par, for example. In those cases, $\mathsf{Base}(K, U)$ will be a category with morphisms reflecting the idea of natural transformations.

For those special cases, we can vary Definition 13 in such a way that a morphism between the two interpretations of K is given by a morphism $\varsigma\colon U \to V$ in Carr and a morphism in $\mathsf{Base}(K, V)$ from $\iota; \varsigma$ to ϱ. We are convinced that all the following constructions and results can be transferred, more or less straightforwardly, to this extended version of morphisms between interpretations. We let this as a topic of future research.

Any context morphism $\varphi\colon K \to G$ induces a functor $\mathtt{Int}(\varphi)\colon \mathtt{Int}(G) \to \mathtt{Int}(K)$ with:

$$\mathtt{Int}(\varphi); \Pi_K = \Pi_G : \mathtt{Int}(G) \to \mathsf{Sem}(\Xi) \tag{3}$$

defined by simple pre-composition: $\mathtt{Int}(\varphi)(\varrho, \mathcal{V}) := (\varphi; \varrho, \mathcal{V})$ for all interpretations $(\varrho, \mathcal{V})$ of G, and for any morphism $\varsigma : (\iota, \mathcal{U}) \to (\varrho, \mathcal{V})$ between two interpretations of G the same underlying morphism $\varsigma : U \to V$ in Carr establishes a morphism $\mathtt{Int}(\varphi)(\varsigma) := \varsigma : (\varphi; \iota, \mathcal{U}) \to (\varphi; \varrho, \mathcal{V})$ between the corresponding two interpretations of K.



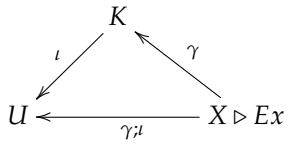
It is straightforward to validate that the assignments $K \mapsto \mathtt{Int}(K)$ and $\varphi \mapsto \mathtt{Int}(\varphi)$ define a functor $\mathtt{Int}\colon \mathsf{Cxt}^{op} \to \mathsf{Cat}$. This is the **model functor** of an Institution of Statements.

3.5.3. Satisfaction Relation and Satisfaction Condition

The last two steps, in establishing an institution, are the definition of *satisfaction relations* and the proof of the so-called *satisfaction condition*. The satisfaction relations are simply given by the semantics of features expressions, as described in Definition 10.

Definition 14 (Satisfaction relation)**.** *For any context* $K \in \mathsf{Cxt}$*, any* $\mathtt{XE}(\Xi)$*-statement* (X, Ex, γ) *in* K *and any interpretation* $(\iota, \mathcal{U})$ *of context* K *we define:*

$$(\iota, \mathcal{U}) \models_K (X, Ex, \gamma) \quad \textit{iff} \quad \gamma;\iota \models^{\mathcal{U}} X \triangleright Ex \quad (\textit{i.e.} \quad \gamma;\iota \in [\![Ex]\!]^{\mathcal{U}}_X) \tag{4}$$



Remark 18 (Validity of Closed Formulas)**.** *In case* $X = K = \mathbf{0}$*, we do have for any* Ξ*-structure* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ *in* $\mathsf{Sem}(\Xi)$ *exactly one interpretation* $(!_U, \mathcal{U})$ *thus for any closed formula* $(\mathbf{0}, Ex, id_{\mathbf{0}})$ *(see Remark 15)* $(!_U, \mathcal{U}) \models_{\mathbf{0}} (\mathbf{0}, Ex, id_{\mathbf{0}})$ *means nothing but that the closed formula* $(\mathbf{0}, Ex, id_{\mathbf{0}})$ *is* valid in $\mathcal{U}$ *in the traditional sense. Therefore, we will also write* $\mathcal{U} \models (\mathbf{0}, Ex, id_{\mathbf{0}})$ *instead of* $(!_U, \mathcal{U}) \models_{\mathbf{0}} (\mathbf{0}, Ex, id_{\mathbf{0}})$*.*

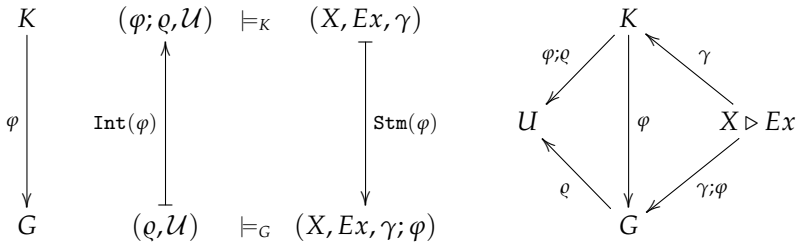
Moreover, the validity of closed formulas is context independent *in the following sense: For any context* K *and any closed expressions* $\mathbf{0} \triangleright Ex$*, we have:*

$$(\iota, \mathcal{U}) \models_K (\mathbf{0}, Ex, !_K) \quad \textit{iff} \quad !_K;\iota = !_U \models^{\mathcal{U}} \mathbf{0} \triangleright Ex \quad \textit{iff} \quad [\![Ex]\!]^{\mathcal{U}}_{\mathbf{0}} = \{!_U\} \quad \textit{iff} \quad \mathcal{U} \models (\mathbf{0}, Ex, id_{\mathbf{0}})$$

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

Corollary 1 (Satisfaction condition)**.** *For any morphism* $\varphi\colon K \to G$ *in* Cxt*, any* $\mathtt{XE}(\Xi)$*-statement* (X, Ex, γ) *in* K *and any interpretation* $(\varrho, \mathcal{U})$ *of context* G *we have:*

$$\mathtt{Int}(\varphi)(\varrho, \mathcal{U}) \models_K (X, Ex, \gamma) \quad \textit{iff} \quad (\varrho, \mathcal{U}) \models_G \mathtt{Stm}(\varphi)(X, Ex, \gamma). \tag{5}$$



Proof. Due to the definition of the functors $\mathtt{Int}\colon \mathsf{Cxt}^{op} \to \mathsf{Cat}$ and $\mathtt{Stm}\colon \mathsf{Cxt} \to \mathsf{Set}$, we obtain the commutative diagram, above on the right, thus the satisfaction condition follows immediately from Definition 14. □

Remark 19 (Satisfaction Condition)**.** *As mentioned in the introductory Section 1.1.6, the finding of* corresponding assignments *and* corresponding evaluations *enabled us to prove in [29] the satisfaction condition for four formalisms in a systematic, uniform and straightforward way. The proof of Corollary 1 mirrors the essence of this uniform and straightforward way at a very high abstraction level.*

Summarizing all definitions and results, we obtain the main result of this section:

Theorem 1 (Institution of Statements)**.** *Any choice of a category* Base*, of subcategories* Var*,* Cxt*,* Carr *of* Base*, of a footprint* Ξ *over* Var*, of a category* $\mathsf{Sem}(\Xi)$ *of* Ξ*-structures and of an* Var_{Obj}*-indexed family* $\mathtt{XE}(\Xi)$ *of first-order* Ξ*-expressions establishes a corresponding* ***Institution of Statements*** $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$.

Remark 20 (Indexed institutions)**.** *We come back to the discussion in Remark 5. If we consider a category of footprints over* Var *we will obtain, due to Theorem 1, for each footprint a corresponding institution of statements. To lift morphisms between footprints to corresponding morphisms between institutions of statements, we have, however, to coordinate somehow the construction of the different institutions (consult Figure 1).*

All institutions should share, besides Base *and* Var *also the same categories* Carr *and* Cxt*. We have to show that this assumption ensures that the assignments* $\Xi \mapsto \mathsf{Str}(\Xi)$ *can be lifted to a functor* Str*. Analogously, the assignments* $\Xi \mapsto \mathtt{FE}(\Xi)$ *should also provide a functor* FE*. Finally, the choices of* $\mathsf{Sem}(\Xi)$ *and* $\mathtt{XE}(\Xi)$ *have to be aligned in such a way that we obtain corresponding restrictions of the functors* Str *and* FE*, respectively.*

Under these assumptions, we will hopefully be able to establish a category of institutions of statements indexed by the category of footprints; thus, we can benefit from all the nice results and constructions in [2]. In particular, the construction of the corresponding Grothendieck institution will surely become relevant.

4. Institutions of Equations

With this section about Institutions of Equational Statements, or short Institutions of Equations, we start to close the circle to the ideas and motivations discussed in the introductory Section 1.1.1 Universal Algebra and Algebraic Specifications. In these areas, substitutions play a central role and, analyzing the situation in these areas, we may obtain also some hints and guidelines for the future development of a more abstract and general account of substitutions in Logics of Statements.

Equations are the main conceptual tool in Universal Algebra. To define equational statements, we could again apply the encapsulation trick we have used in the last subsections to define statements for footprints with feature symbols only. That is, we could introduce atomic equations $X \triangleright t_1 = t_2$, define atomic equational statements $(X, t_1 = t_2, \gamma)$ in contexts K with $\gamma : X \to K$ and translate atomic equational statements along context morphisms by simple post-composition.

This idea works fine as long as we are only interested in formalisms to describe and specify algebraic structures. The encapsulation approach seems to be not appropriate, however, to describe and work with instances of equations w.r.t. substitutions of variables by terms. The construction of those instances is a crucial tool in any deduction calculus in Universal Algebra; thus, we decided to work instead of the encapsulation-based two-step approach with a one-step approach defining directly equations $K \triangleright t_1 = t_2$ in contexts K.

This means that we adapt for Institutions of Equations the construction scheme in Figure 1 in the following way: We have $\mathsf{Str}(\Xi) = \mathsf{Sem}(\Xi)$. Step (6) is dropped and we construct directly $\mathtt{Stm}(K)$ as a set of equations in context K. Correspondingly, the satisfaction relations $\models_K$ are defined by means of the evaluation of terms in algebras.

As a complement to the FOL-example, we consider many-sorted total algebras and conditional equations. In this section, we define corresponding Institutions of Equations while conditional equations are formalized and discussed in Section 5.3.

In accordance with the FOL-example, we fix a finite set $S \in \mathsf{Set}_{Obj}$ of sort symbols and choose as Base_{EQ} the interpretation category $\mathsf{Set}^S = [S \to \mathsf{Set}]$. Var_{EQ} is the full subcategory of Base_{EQ} given by all finite and disjoint S-sets $X = (X_s \mid s \in S)$ with X_s a subset of the set $\{x, x_1, x_2, \ldots, y, y_1, y_2, \ldots\}$ for all $s \in S$.

4.1. Signatures, Algebras and Contexts

Signatures $\Sigma = (\Omega, in, out)$ correspond to traditional many-sorted algebraic signatures and are given by a set Ω of operation symbols, a map in assigning to each operation symbol $\omega \in \Omega$ an object $in(\omega)$ in Var_{EQ}, its **arity**, and a map $out : \Omega \to S$. For convenience, we assume that $\bigcup in(\omega) = \{x_1, x_2, \ldots, x_n\}$, $n \geq 0$; thus, we can represent $in(\omega)$ as a list $[x_1{:}\, s_1, x_2{:}\, s_2, \ldots, x_n{:}\, s_n]$ of variable declarations (compare Example 11).

We have $\mathsf{Carr}_{EQ} := \mathsf{Base}_{EQ}$. As structures, we consider **Σ- algebras** $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ with an S-set $A = (A_s \mid s \in S)$ and a family $\Omega^{\mathcal{A}}$ of operations $\omega^{\mathcal{A}} : A^{in(\omega)} \to A_{out(\omega)}$, $\omega \in \Omega$, where $A^{in(\omega)}$ is a shorthand for the set $\mathsf{Set}^S(in(\omega), A)$ of all S-maps from $in(\omega)$ into A.

A **homomorphism** $\varsigma : \mathcal{A} \to \mathcal{B}$ between Σ-algebras $\mathcal{A}$ and $\mathcal{B}$ is given by an S-map $\varsigma = (\varsigma_s \mid s \in S) : A \to B$ such that $\omega^{\mathcal{A}}; \varsigma_s = \varsigma^{in(\omega)}; \omega^{\mathcal{B}}$ for all $\omega \in \Omega$ where the map $\varsigma^{in(\omega)} : A^{in(\omega)} \to B^{in(\omega)}$ is defined by $\varsigma^{in(\omega)}(\tau) := \tau; \varsigma$ for all S-maps $\tau \in A^{in(\omega)}$.

$$\begin{array}{ccc} A^{in(\omega)} & \xrightarrow{\omega^{\mathcal{A}}} & A_s \\ {\scriptstyle \varsigma^{in(\omega)}}\downarrow & = & \downarrow{\scriptstyle \varsigma_s} \\ B^{in(\omega)} & \xrightarrow{\omega^{\mathcal{A}}} & B_s \end{array}$$

$\mathsf{Alg}(\Sigma)$ is the category of all Σ-algebras and all homomorphisms between them.

We choose $\mathsf{Cxt}_{EQ} := \mathsf{Carr}_{EQ} = \mathsf{Base}_{EQ} = \mathsf{Set}^S$. The model functor of an Institution of Equations is defined in full analogy to Institutions of Statements.

An interpretation $(\iota, \mathcal{A})$ of a context K in Cxt_{EQ}, i.e., of an S-set K, is given by a Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ and an S-map $\iota{:}\, K \to A$.

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

$$\begin{array}{ccccc} & & K & & \\ & {\scriptstyle \iota}\swarrow & = & \searrow{\scriptstyle \varrho} & \\ A & & \xrightarrow{\varsigma} & & B \end{array}$$

For any context K in Cxt_{EQ}, $\mathtt{Int}(K)$ denotes the category of all interpretations of K and all morphisms between them and $\Pi_K : \mathtt{Int}(K) \to \mathsf{Alg}(\Sigma)$ is the corresponding projection functor. The fiber over a Σ-algebra $\mathcal{A}$ represents the semantics of a context K in $\mathcal{A}$, i.e., the set $A^K := \mathsf{Set}^S(K, A)$ of all S-maps from K into the carrier of $\mathcal{A}$.

Note that, in the case of the empty S-set $K = \mathbf{0} = (\emptyset \mid s \in S)$ the projection functor $\Pi_{\mathbf{0}} : \mathtt{Int}(\mathbf{0}) \to \mathsf{Alg}(\Sigma)$ is an isomorphism.

Any S-map $\varphi{:}\, K \to G$ induces a functor $\mathtt{Int}(\varphi){:}\, \mathtt{Int}(G) \to \mathtt{Int}(K)$ with:

$$\mathtt{Int}(\varphi); \Pi_K = \Pi_G : \mathtt{Int}(G) \to \mathsf{Alg}(\Sigma) \tag{6}$$

defined by simple pre-composition: $\mathtt{Int}(\varphi)(\varrho, \mathcal{B}) := (\varphi; \varrho, \mathcal{B})$ for all interpretations $(\varrho, \mathcal{B})$ of G, and for any morphism $\varsigma{:}\, (\iota, \mathcal{A}) \to (\varrho, \mathcal{B})$ between two interpretations of G the same underlying S-map $\varsigma{:}\, A \to B$ establishes a morphism $\mathtt{Int}(\varphi)(\varsigma) := \varsigma{:}\, (\varphi; \iota, \mathcal{A}) \to (\varphi; \varrho, \mathcal{B})$ between the corresponding two interpretations of K.

$$\begin{array}{ccc} K & \xrightarrow{\varphi} & G \\ & {\scriptstyle \varphi;\varrho}\searrow \quad \swarrow{\scriptstyle \varrho} & \\ & B & \end{array} \qquad \begin{array}{ccc} K & \xrightarrow{\varphi} & G \\ {\scriptstyle \varphi;\iota}\downarrow & {\scriptstyle \varphi;\varrho}\quad {\scriptstyle \iota} & \downarrow{\scriptstyle \varrho} \\ A & \xrightarrow{\varsigma} & B \end{array} \qquad \begin{array}{ccc} \mathtt{Int}(K) & \xleftarrow{\mathtt{Int}(\varphi)} & \mathtt{Int}(G) \\ & {\scriptstyle \Pi_K}\searrow \quad = \quad \swarrow{\scriptstyle \Pi_G} & \\ & \mathsf{Alg}(\Sigma) & \end{array}$$

The assignments $K \mapsto \mathtt{Int}(K)$ and $\varphi \mapsto \mathtt{Int}(\varphi)$ define a functor $\mathtt{Int} : \mathsf{Cxt}_{EQ}^{op} \to \mathsf{Cat}$. This is the **model functor** of an Institution of Equations.

4.2. Terms and Equations

To define equations, we need terms! For any S-set K the S-set $T_\Sigma(K)$ of all **Σ-terms on** K is defined inductively as the smallest S-set such that:

1. $K \subseteq T_\Sigma(K)$
2. $\omega\langle\rangle \in T_\Sigma(K)_{out(\omega)}$ for all $\omega \in \Omega$ with $in(\omega)$ the empty S-set $\mathbf{0} = (\emptyset \mid s \in S)$.
3. $\omega\langle\tau_{s_1}(x_1), \ldots, \tau_{s_n}(x_n)\rangle \in T_\Sigma(K)_{out(\omega)}$ for all $\omega \in \Omega$ with $in(\omega)$ non-empty and all S-maps $\tau : in(\omega) \to T_\Sigma(K)$ where $[x_1{:}\, s_1, x_2{:}\, s_2, \ldots, x_n{:}\, s_n]$ is the assumed representation of $in(\omega)$ as a list of variable declarations.

A **Σ-equation** $(K, t_1 = t_2)$ **in** K is given by two Σ-terms $t_1, t_2 \in T_\Sigma(K)_s$ for some $s \in S$ and $\mathtt{Eq}(K)$ denotes the set of all Σ-equations $(K, t_1 = t_2)$ in K. In the usual way, the inductive definition of Σ-terms allows us to extend any S-map $\varphi{:}\, K \to G$ between S-sets to an S-map $\varphi^* : T_\Sigma(K) \to T_\Sigma(G)$ such that $\subseteq; \varphi^* = \varphi; \subseteq$ thus $\varphi{:}\, K \to G$ induces a map $\mathtt{Eq}(\varphi) : \mathtt{Eq}(K) \to \mathtt{Eq}(G)$ with:

$$\mathtt{Eq}(\varphi)(K, t_1 = t_2) := (G, \varphi^*(t_1) = \varphi^*(t_2)) \tag{7}$$

for all Σ-equations $(K, t_1 = t_2)$ in K.

$$\begin{array}{ccc} T_\Sigma(K) & \xrightarrow{\varphi^*} & T_\Sigma(G) \\ {\scriptstyle\subseteq}\uparrow & & \uparrow{\scriptstyle\subseteq} \\ K & \xrightarrow{\varphi} & G \end{array}$$

Since $id_K^* = id_{T_\Sigma(K)}$ and $(\varphi; \psi)^* = \varphi^*; \psi^*$ for all $\varphi{:}\, K \to G$, $\psi{:}\, G \to H$, the assignments $K \mapsto \mathtt{Eq}(K)$ and $\varphi \mapsto \mathtt{Eq}(\varphi)$ define a functor $\mathtt{Eq} : \mathsf{Cxt}_{EQ} \to \mathsf{Set}$. This is the **sentence functor** of an Institution of Equations.

The semantics of terms is based on the **evaluation of terms** in algebras: Due to the inductive definition of Σ-terms, we can extend any interpretation $\iota{:}\, K \to A$ of a context K in a Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ to an S-map $\iota^\circ : T_\Sigma(K) \to A$ such that:

$$\subseteq; \iota^\circ = \iota. \tag{8}$$

$$\begin{array}{ccc} K & \xrightarrow{\subseteq} & T_\Sigma(K) \\ & {\scriptstyle\iota}\searrow & \downarrow{\scriptstyle\iota^\circ} \\ & & A \end{array}$$

In such a way, we can define the semantics $t^{\mathcal{A}}$ of a Σ-term $t \in T_\Sigma(K)$, $s \in S$ in a Σ-algebra $\mathcal{A}$ as a map $t^{\mathcal{A}} : A^K \to A_s$ defined by $t^{\mathcal{A}}(\iota) := \iota^\circ(t)$ for all $\iota{:}\, K \to A$. Thus, feature expressions represent derived properties while terms represent *derived operations*!

4.3. Satisfaction Relation and Satisfaction Condition

Definition 15 (Satisfaction relation for equations)**.** *For any context $K \in \mathsf{Cxt}_{EQ}$, any Σ-equation $(K, t_1 = t_2)$ in K and any interpretation $(\iota, \mathcal{A})$ of context K in a Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$, we define:*

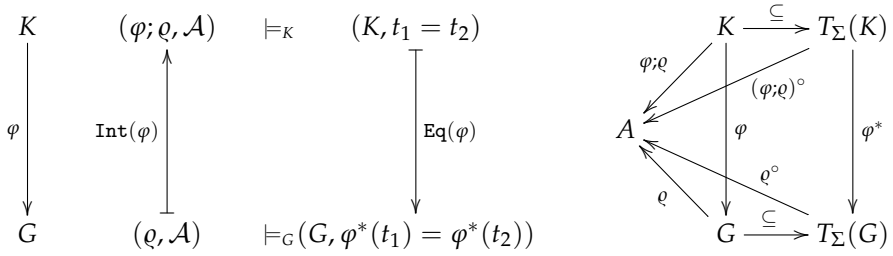
$$(\iota, \mathcal{A}) \models_K (K, t_1 = t_2) \quad \textit{iff} \quad \iota^\circ(t_1) = \iota^\circ(t_2) \quad (\textit{i.e.} \quad t_1^{\mathcal{A}}(\iota) = t_2^{\mathcal{A}}(\iota)\ ) \tag{9}$$

$$\begin{array}{ccc} K & \xrightarrow{\subseteq} & T_\Sigma(K) \\ & {\scriptstyle\iota}\searrow & \swarrow{\scriptstyle\iota^\circ} \\ & A & \end{array}$$

The satisfaction condition is ensured by the well-behaved interplay of translations of terms along context morphisms and evaluations of terms.

Proposition 1 (Satisfaction condition for equations)**.** *For any morphism* $\varphi\colon K \to G$ *in* Cxt_{EQ}*, any* Σ*-equation* $(K, t_1 = t_2)$ *in* K *and any interpretation* $(\varrho, \mathcal{A})$ *of context* G *in a* Σ*-algebra* $\mathcal{A} = (A, \Omega^{\mathcal{A}})$*, we have:*

$$\mathtt{Int}(\varphi)(\varrho, \mathcal{A}) \models_K (K, t_1 = t_2) \quad \textit{iff} \quad (\varrho, \mathcal{A}) \models_G \mathtt{Eq}(\varphi)(K, t_1 = t_2). \tag{10}$$



Proof. Due to the definition of the functors $\mathtt{Int}\colon \mathsf{Cxt}_{EQ}^{op} \to \mathsf{Cat}$, $\mathtt{Eq}\colon \mathsf{Cxt}_{EQ} \to \mathsf{Set}$ and the fact that $(\varphi;\varrho)^\circ = \varphi^*;\varrho^\circ$, we obtain the commutative diagram, above on the right, thus the satisfaction condition follows immediately from Definition 15. □

Summarizing all definitions and results, we obtain the main result in this section:

Proposition 2 (Institution of Equations)**.** *Any choice of a finite set* S *and a signature* $\Sigma = (\Omega, in, out)$ *establishes a corresponding* ***Institution of Equations*** $\mathcal{IE} = (\mathsf{Cxt}_{EQ}, \mathtt{Eq}, \mathtt{Int}, \models)$.

5. Sketches

Any institution gives us a corresponding category of presentations and an extension of the model functor of the institution to the category of presentations at hand [2,31]. We outline this construction for Institutions of Statements and Institutions of Equations.

We would like to use a specific term to distinguish presentations for Institutions of Statements or Equations, resp., from presentations in general. Since many of our motivating examples are variants of sketches, we will simply use the term *sketch*. In Sections 5.1 and 5.2, we concentrate on sketches for Institutions of Statements while Section 5.3 outlines the corresponding variations for Institutions of Equations.

5.1. Sketches of Statements: Syntax and Semantics

To be prepared for the topics in Section 6, we introduce a very abstract and semantics-independent concept of *sketch*.

Definition 16 (Sketch)**.** *Let us have a category* Ct *of* contexts *and a functor* $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$*, assigning to each* $K \in \mathsf{Ct}_{Obj}$ *a set* $\mathtt{St}(K)$ *of all statements in context* K*.*

An $\mathtt{St}$***- sketch*** $\mathbb{K} = (K, St^{\mathbb{K}})$ *is given by a context* $K \in \mathsf{Ct}_{Obj}$ *and a set* $St^{\mathbb{K}} \subseteq \mathtt{St}(K)$ *of statements in context* K*.*

In this subsection, we consider the case $\mathsf{Ct} = \mathsf{Cxt}$, $\mathtt{St} = \mathtt{Stm}$ with $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ an arbitrary Institution of Statements according to Theorem 1.

All definitions and constructions are, however, institution-independent; thus, they apply analogously to the case $\mathsf{Ct} = \mathsf{Cxt}_{EQ}$, $\mathtt{St} = \mathtt{Eq}$ with $\mathcal{IE} = (\mathsf{Cxt}_{EQ}, \mathtt{Eq}, \mathtt{Int}, \models)$ an arbitrary Institution of Equations according to Proposition 2.

Example 36 (FOL: Sketches)**.** *We extend the sample context* K *in Example 31 to an* $\mathtt{Stm}_{FOL}$*-sketch* $\mathbb{K}$ *with the atomic statements (facts)* $\mathtt{parent}(Anna, Uwe, Gabi)$, $\mathtt{parent}(Uwe, Heinz, Dora)$, $\mathtt{male}(Michael)$ *and the proper first-order statements* $([y : prs], sbl, (y \mapsto Michael))$,

$([y : prs], sbl, (y \mapsto Uwe))$, $([y : prs, x : nat], younger, (y \mapsto Michael, x \mapsto 12))$, $([y : prs], sbl, (y \mapsto Gabi))$. *The expression* $[y : prs] \triangleright sbl$, *representing the property* being sibling of someone, *and the expression* $[y : prs, x : nat] \triangleright younger$, *representing the property* younger than, *are defined in Example 26.*

Example 37 (ALC: Sketches)**.** *Contexts in ALC are* sets N_O of individual names *as already mentioned in Example 32. A* concept assertion *in ALC, i.e., a statement of the form* $a : \mathtt{C}$ *with* $a \in N_O$ *and* $\mathtt{C}$ *a (derived) concept, can be seen as a statement* $(\{x_1\}, \mathtt{C}(x_1), (x_1 \mapsto a))$ *in* N_O *where the assignment* $(x_1 \mapsto a)$ *defines a binding* $\beta : \{x_1\} \to N_O$ *with* $\beta(x_1) = a$.

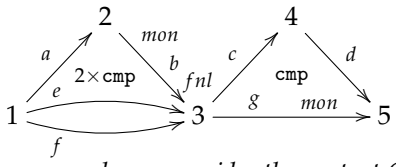
A role assertion, *i.e., a statement of the form* $(a, b) : \mathtt{R}$ *where* $a, b \in N_O$ *and* $\mathtt{R}$ *is a role, can be seen as a statement* $(\{x_1, x_2\}, \mathtt{R}(x_1, x_2), (x_1 \mapsto a, x_2 \mapsto b))$ *in* N_O. *An* ABox *in ALC is a finite set of* assertional axioms. *Thus, a pair* $(N_O, \mathcal{A})$ *of a* set N_O of individual names *and an ABox* $\mathcal{A}$ *of* assertional axioms *in* N_O *is just an* $\mathtt{Stm}_{ALC}$*-sketch in our sense.*

Example 38 (mFOL: Sketches)**.** *We extend the context* $(K, \tau_K : K \to M_{mF})$ *in Example 33 to an* $\mathtt{Stm}_{mF}$*-sketch with the atomic statements* $\mathtt{un}(male : \boldsymbol{P}, prs : \boldsymbol{S})$, $\mathtt{bin}(less : \boldsymbol{P}, nat : \boldsymbol{S}, nat : \boldsymbol{S})$, $\mathtt{bin}(age : \boldsymbol{P}, prs : \boldsymbol{S}, nat : \boldsymbol{S})$ *and* $\mathtt{trt}(parent : \boldsymbol{P}, prs : \boldsymbol{S}, prs : \boldsymbol{S}, prs : \boldsymbol{S})$.

Obviously, this $\mathtt{Stm}_{mF}$*-sketch describes exactly the sample footprint* Ξ_{FOL} *in Example 11! Actually, we can describe all FOL-footprints, declaring only unary, binary or tertiary predicate symbols, as* $\mathtt{Stm}_{mF}$*-sketches. This fact confirms that the mFOL-example establishes indeed a meta-level for the FOL-example.*

We have to be aware, however, that not all $\mathtt{Stm}_{mF}$*-sketches correspond to FOL-footprints. For each predicate symbol in a FOL-footprint, we have to declare an arity, and this arity should be unique! Therefore, only those* $\mathtt{Stm}_{mF}$*-sketches, with exactly one atomic statement for each element in* $\tau_K^{-1}(\boldsymbol{P})$ *correspond to FOL-footprints. To describe those requirements concerning the structure of sketches, we can utilize* sketch implications, *introduced in the next subsection, and/or* sketch constraints *introduced in Section 6.*

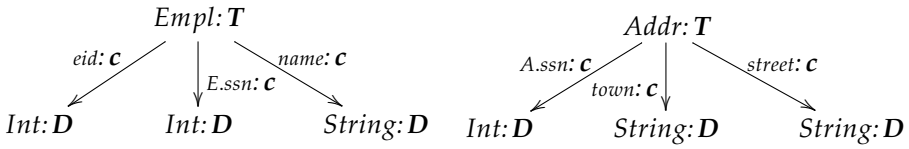
Example 39 (CT: Sketches)**.** *These are just the sketches, as we know them from Category Theory, with the essential difference that we are not restricting ourselves to commutative, limit and colimit statements only. We do not need to encode, for example, the concept* monomorphism *by means of pullbacks but can define it directly as a property of edges utilizing the* Ξ_{CT}*-expressions we discussed in Example 29.*



As an example, we consider the context G from Example 34 and extend it to an $\mathtt{Stm}_{CT}$*-sketch* $\mathbb{G} = (G, St^{\mathbb{G}})$ *visualized above.* $St^{\mathbb{G}}$ *contains the atomic statements* $\mathtt{cmp}(a, b, e)$, $\mathtt{cmp}(a, b, f)$, $\mathtt{cmp}(c, d, g)$ *and the proper first-order statements:*

$(xv, fnl, (xv \mapsto 3))$, $(xv_1 \xrightarrow{xe} xv_2, mon, (xe \mapsto b))$, $(xv_1 \xrightarrow{xe} xv_2, mon, (xe \mapsto g))$.

Example 40 (RM: Sketches)**.** *We extend the sample context* (K, τ_K) *from Example 35 to an* $\mathtt{Stm}_{RM}$*-sketch* $\mathbb{K} = (K, St^{\mathbb{K}})$. *First, we declare two tables, i.e.,* $St^{\mathbb{K}}$ *contains two atomic statements* $\mathtt{tb}(3)(\gamma_1)$, $\mathtt{tb}(3)(\gamma_2)$ *with bindings* γ_1 *and* γ_2 *visualized by the following typed graphs:*



Then, we declare for each table a primary key, i.e., we add two atomic statements $\mathtt{pk}(\gamma_3)$, $\mathtt{pk}(\gamma_4)$ *with bindings* γ_3, γ_4 *given by* $Empl{:}\,\boldsymbol{T} \xrightarrow{eid{:}\,\boldsymbol{C}} Int{:}\,\boldsymbol{D}$ *and* $Addr{:}\,\boldsymbol{T} \xrightarrow{A.ssn{:}\,\boldsymbol{C}} Int{:}\,\boldsymbol{D}$ *. Moreover, we declare a foreign key* $\mathtt{fk}(\gamma_5)$ *with* γ_5 *depicted by* $Empl{:}\,\boldsymbol{T} \xrightarrow{E.ssn{:}\,\boldsymbol{C}} Int{:}\,\boldsymbol{D} \xleftarrow{A.ssn{:}\,\boldsymbol{C}} xt_2{:}\,\boldsymbol{T}$ *.*

We could also require that each employee has a name and add an atomic statement $\mathtt{tot}(\gamma_6)$ *with* γ_6 *given by* $Empl{:}\,\boldsymbol{T} \xrightarrow{name{:}\,\boldsymbol{C}} String{:}\,\boldsymbol{D}$ *.*

Analogously to the requirements in Example 38, we do have the requirement that a table identifier can only appear once in a $\mathtt{tb(n)}$*-statement. There are, however, other database specific requirements: Any table should have exactly one primary key, a foreign key has to refer to a primary key, and others. As said before, to describe those kinds of structural requirements, we need* sketch implications *and/or* sketch constraints. □

For any context K in Cxt, any set $S \subseteq \mathtt{Stm}(K)$ of statements in K and any interpretation $(\iota, \mathcal{U})$ of context K in a Ξ-structure $\mathcal{U}$ we define, relying on Definition 14:

$$(\iota, \mathcal{U}) \models_K S \quad \text{iff} \quad (\iota, \mathcal{U}) \models_K (X, Ex, \gamma) \ \text{ for all } (X, Ex, \gamma) \in S. \tag{11}$$

Be aware that the statements in S may have different variable declarations X.

Definition 17 (Interpretation of sketch)**.** *A* ***valid interpretation (model)*** *of an* $\mathtt{Stm}$*-sketch* $\mathbb{K} = (K, St^{\mathbb{K}})$ *is an interpretation* $(\iota, \mathcal{U})$ *of context* K *such that* $(\iota, \mathcal{U}) \models_K St^{\mathbb{K}}$*.*

We denote by $\mathtt{Int}(\mathbb{K})$ *the full subcategory of* $\mathtt{Int}(K)$ *determined by all valid interpretations of* $\mathbb{K}$ *and by* $\Pi_{\mathbb{K}} : \mathtt{Int}(\mathbb{K}) \to \mathsf{Sem}(\Xi)$ *we denote the corresponding restriction of the projection functor* $\Pi_K : \mathtt{Int}(K) \to \mathsf{Sem}(\Xi)$*.*

Remark 21 (Traditional presentations)**.** *If* Base *has an initial object* $\mathbf{0}$*, we can consider sketches* $(\mathbf{0}, St)$ *with* St *only containing* closed formulas*, i.e., statements of the form* $(\mathbf{0}, Ex, id_{\mathbf{0}})$ *(see Remark 15). As discussed before, the projection functor* $\Pi_{\mathbf{0}} : \mathtt{Int}(\mathbf{0}) \to \mathsf{Sem}(\Xi)$*, due to Definition 13, is an isomorphism. In such a way,* $(\mathbf{0}, St)$ *is not only determining the interpretation subcategory* $\mathtt{Int}(\mathbf{0}, St) \sqsubseteq \mathtt{Int}(\mathbf{0})$ *but can also be seen as a* presentation (specification) *of the corresponding full subcategory* $\mathsf{Sem}(\Xi, (\mathbf{0}, St)) := \Pi_{\mathbf{0}}(\mathtt{Int}(\mathbf{0}, St))$ *of* $\mathsf{Sem}(\Xi)$ *isomorphic to* $\mathtt{Int}(\mathbf{0}, St)$*.*

In other words: due to Remark 18, we can describe $\mathsf{Sem}(\Xi, (\mathbf{0}, St))$ *as the full subcategory of* $\mathsf{Sem}(\Xi)$ *given by all* Ξ*-structures* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ *in* $\mathsf{Sem}(\Xi)$ *such that* $\mathcal{U} \models (\mathbf{0}, Ex, id_{\mathbf{0}})$ *for all closed formulas* $(\mathbf{0}, Ex, id_{\mathbf{0}})$ *in* St*.*

Example 41 (FOL: Interpretations)**.** *If we interpret the symbolic literals in* $K_{prs} = \{Anna, Michael, Dora, Heinz, Sorin, Gabi, Uwe\}$ *by the real persons in our family in December 2021, we will not obtain a valid interpretation of the* $\mathtt{Stm}_{FOL}$*-sketch* $\mathbb{K}$ *in Example 36 since the statement* $([y : prs], sbl, (y \mapsto Gabi))$ *is not satisfied by this interpretation. If we use, however, the statement* $([y : prs], \neg sbl, (y \mapsto Gabi))$ *instead, the interpretation becomes valid.*

Note that the statement $([y : prs], sbl, (y \mapsto Uwe))$ *is satisfied by the interpretation even if there is no witness for this statement in the context. Uwe's only sister Brita is not present in the context K!*

Example 42 (mFOL: Interpretations)**.** *An interpretation of the sample context* (K, τ_K) *assigns to each element in* $\tau_K^{-1}(\boldsymbol{S}) := \{prs, nat\}$ *and* $\tau_K^{-1}(\boldsymbol{P}) := \{\mathtt{parent}, \mathtt{male}, \mathtt{age}, \mathtt{less}\}$*, respectively, a set. Since* Base_{mF} *is the slice category* SET/M_{mF}*, a certain set can either serve as a sort or as a predicate.*

Our choice of $\mathsf{Sem}(\Xi_{mF})$ *in Example 23 ensures, in addition, that the valid interpretations of the sample* $\mathtt{Stm}_{mF}$*-sketch from Example 38 are in one-to-one correspondence to the* Ξ_{FOL}*-structures in* $\mathsf{Sem}(\Xi_{FOL}) = \mathsf{Str}(\Xi_{FOL})$*. We do have such a semantical one-to-one correspondence for any FOL-footprint, declaring only unary, binary or tertiary predicate symbols, and the corresponding*

$\mathtt{Stm}_{mF}$*-sketch. This confirms that the mFOL-example establishes a meta-level for the FOL-example also w.r.t. semantics.*

It is maybe worth mentioning that any $\mathtt{Stm}_{mF}$*-sketch with two different atomic statements for, at least, one element in* $\tau_K^{-1}(\boldsymbol{P})$ *has no valid interpretation at all in* $\mathsf{Sem}(\Xi_{mF})$.

Example 43 (Category Theory: Interpretations)**.** *Since we defined in Example 24 a very liberal semantics, we do have interpretations* $(\iota, \mathcal{U})$, $\mathcal{U} = (U, \Phi^{\mathcal{U}}_{CT})$ *of the sample* $\mathtt{Stm}_{CT}$*-sketch* $\mathbb{G} = (G, St^{\mathbb{G}})$ *in Example 39, where the graph homomorphism* $\iota : G \to U$ *maps the edges* e *and* f *to different edges in* U *even if both are declared as the composition of* a *and* b.

If we would have also included into the sketch $\mathbb{G}$ *the* general statement $(\mathbf{0}, guc, !_G)$ *with the closed expression* $\mathbf{0} \triangleright guc$ *expressing the property* composititon is always unique *(see Example 29),* $(\iota, \mathcal{U})$ *could be only a valid interpretation of* $\mathbb{G}$ *if* ι *identifies* e *and* f.

Example 44 (RM: Interpretations)**.** *Analogously to Example 42, our choice of* $\mathsf{Sem}(\Xi_{RM})$ *in Example 25 ensures that the valid interpretations of "well-formed"* $\mathtt{Stm}_{RM}$*-sketches, i.e.,* $\mathtt{Stm}_{RM}$*-sketches representing database schemata, formalize exactly the traditional semantics of database schemata as outlined in Subsection 3.1.5.*

Morphisms between sketches are defined by means of semantic entailment in a certain Institution of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$.

Definition 18 (Statement entailment)**.** *For any context* K *in* Cxt *and any sets* $S, T \subseteq \mathtt{Stm}(K)$ *of statements in* K*, we say that* S ***entails*** T ***in a*** Ξ***- structure*** $\mathcal{U}$, $S \Vdash^{\mathcal{U}}_K T$ *in symbols, if, and only if, for all interpretations* $(\iota, \mathcal{U})$ *of* K *in* $\mathcal{U}$: $(\iota, \mathcal{U}) \models_K S$ *implies* $(\iota, \mathcal{U}) \models_K T$.

S ***entails*** T, $S \Vdash_K T$ *in symbols, if, and only if,* $S \Vdash^{\mathcal{U}}_K T$ *for all* Ξ*-structures* $\mathcal{U}$ *in* $\mathsf{Sem}(\Xi)$.

Definition 19 (Sketch morphism)**.** *An* $\mathcal{IS}$***-morphism*** $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ *between two* $\mathtt{Stm}$*-sketches* $\mathbb{K} = (K, St^{\mathbb{K}})$, $\mathbb{G} = (G, St^{\mathbb{G}})$ *is a morphism* $\varphi : K \to G$ *in* Cxt *such that* $St^{\mathbb{G}} \Vdash_G \mathtt{Stm}(\varphi)(St^{\mathbb{K}})$. *An* $\mathcal{IS}$*-morphism* $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ *is called* ***strict*** *if* $St^{\mathbb{G}} \supseteq \mathtt{Stm}(\varphi)(St^{\mathbb{K}})$.

$\mathsf{Sk}(\mathcal{IS})^m$ *denotes the category of all* $\mathtt{Stm}$*-sketches and all* $\mathcal{IS}$*-morphisms between them. Its subcategory of all* $\mathtt{Stm}$*-sketches and all strict* $\mathcal{IS}$*-morphisms is denoted by* $\mathsf{Sk}(\mathcal{IS})^m_s$.

We will consider three different kinds of directed relationships between sketches distinguished by three different kinds of arrow-symbols. We choose the arrow-symbol "$\dashrightarrow$" for sketch morphisms since it is the kind of directed relationship we will mention the least.

If $\mathcal{IS}$ is clear from the context, we will also use the shorthand notations Sk^m and Sk^m_s instead of $\mathsf{Sk}(\mathcal{IS})^m$ and $\mathsf{Sk}(\mathcal{IS})^m_s$, respectively.

$\mathcal{IS}$-morphisms $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ with $K = G$ and $\varphi = id_K$ simply reflect statement entailments. For any $\mathcal{IS}$-morphism $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$, the condition $St^{\mathbb{G}} \Vdash_G \mathtt{Stm}(\varphi)(St^{\mathbb{K}})$ ensures, due to the satisfaction condition that the functor $\mathtt{Int}(\varphi) : \mathtt{Int}(G) \to \mathtt{Int}(K)$ restricts to a functor from $\mathtt{Int}(\mathbb{G})$ into $\mathtt{Int}(\mathbb{K})$. In such a way, the assignments $\mathbb{K} \mapsto \mathtt{Int}(\mathbb{K})$ extend to a functor $\mathtt{Int}^{\mathsf{Sk}} : (\mathsf{Sk}^m)^{op} \to \mathsf{Cat}$.

According to well-known general results (see Corollary 4.3 in [2]), we know that Sk^m has whatever limits or colimits the category Cxt has since limits and colimits in the category Sk^m of $\mathtt{Stm}$-sketches and $\mathcal{IS}$-morphisms are constructed by means of limits and colimits in the category Cxt, respectively (compare Propositions 5 and 6). This ensures also that we do have *amalgamation* [1,2,38]: $\mathtt{Int}^{\mathsf{Sk}} : (\mathsf{Sk}^m)^{op} \to \mathsf{Cat}$ maps all colimits in Cxt that are preserved by the inclusion $\mathsf{Cxt} \sqsubseteq \mathsf{Base}$, to limits in Cat.

The Theory of Institutions gives us "for free" $\mathtt{Stm}$-sketches, $\mathcal{IS}$-morphisms, the category Sk^m as well as the *extended model functor* $\mathtt{Int}^{\mathsf{Sk}} : (\mathsf{Sk}^m)^{op} \to \mathsf{Cat}$.

However, to employ sketches as a specification formalism and to develop deduction calculi for sketches, we need a number of other concepts, constructions and results.

5.2. Sketches of Statements vs. Structures

In the Introduction, we discussed, among other things, two central motivations for the development of our framework: (1) We want to be able to give a general abstract account of the concept of free structures generalizing concepts like a group generated by a set of generators and a set of defining relations. (2) We want to provide an alternative general mechanism to encode structures "syntactically" that avoids the kind of circularity inherent to the technique of "signature extensions".

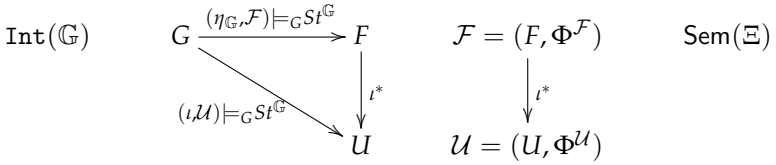
In the remaining part of this section, we outline proposals to meet these objectives.

5.2.1. Freely Generated Structures

To reconstruct the concept of a group generated by a set of generators and a set of defining relations, we need operations only. Those cases of free algebras are discussed in Section 5.3.1.

First, We Consider Structures Freely Generated in $\mathsf{Sem}(\Xi)$

A Ξ-structure $\mathcal{F} = (F, \Phi^{\mathcal{F}})$ is freely generated in $\mathsf{Sem}(\Xi)$ by an $\mathtt{Stm}$-sketch $\mathbb{G} = (G, St^{\mathbb{G}})$ if, and only if, $\mathcal{F}$ is in $\mathsf{Sem}(\Xi)$ and there is a *valid interpretation* $(\eta_{\mathbb{G}}, \mathcal{F})$ of $\mathbb{G}$ in $\mathcal{F}$ that is *universal relative to* $\mathsf{Sem}(\Xi)$. That is, for all Ξ-structures $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ in $\mathsf{Sem}(\Xi)$ and all valid interpretations $(\iota, \mathcal{U})$ of $\mathbb{G}$ in $\mathcal{U}$ there exists a unique morphism $\iota^* : \mathcal{F} \to \mathcal{U}$ in $\mathsf{Sem}(\Xi)$ such that $\eta_{\mathbb{G}}; \iota^* = \iota$ in Base, i.e., such that ι^* establishes an interpretation morphism $\iota^* : (\eta_{\mathbb{G}}, \mathcal{F}) \to (\iota, \mathcal{U})$ in $\mathtt{Int}(\mathbb{G}) \sqsubseteq \mathtt{Int}(G)$ according to Definition 13.



A Ξ-structure, freely generated in $\mathsf{Sem}(\Xi)$ by an $\mathtt{Stm}$-sketch $\mathbb{G} = (G, St^{\mathbb{G}})$, is obviously uniquely determined "up to isomorphism in $\mathsf{Sem}(\Xi)$" if it exists.

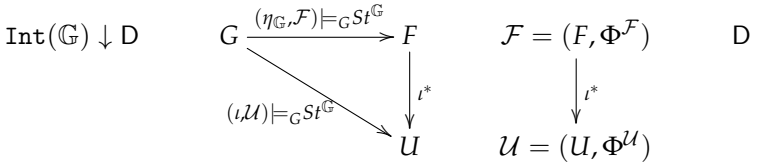
The universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$ entails that $(\eta_{\mathbb{G}}, \mathcal{F})$ is initial in $\mathtt{Int}(\mathbb{G})$, thus the projection functor $\Pi_{\mathbb{G}} : \mathtt{Int}(\mathbb{G}) \to \mathsf{Sem}(\Xi)$ establishes a functor from $\mathtt{Int}(\mathbb{G})$ into the co-slice category $\mathcal{F}/\mathsf{Sem}(\Xi)$.

In the case that $St^{\mathbb{G}}$ contains only atomic statements, the definition of morphisms between Ξ-structures ensures $(\eta_{\mathbb{G}}; \varrho, \mathcal{U}) \models_G St^{\mathbb{G}}$ for any morphism $\varrho : \mathcal{F} \to \mathcal{U}$ in $\mathsf{Sem}(\Xi)$; thus, the assignments $(\varrho : \mathcal{F} \to \mathcal{U}) \mapsto (\eta_{\mathbb{G}}; \varrho, \mathcal{U})$ establish a functor from $\mathcal{F}/\mathsf{Sem}(\Xi)$ into $\mathtt{Int}(\mathbb{G})$. Due to the universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$, we obtain $(\eta_{\mathbb{G}}; \varrho)^* = \varrho$. Together with the equation $\eta_{\mathbb{G}}; \iota^* = \iota$, this ensures that the two functors establish an isomorphism between $\mathtt{Int}(\mathbb{G})$ and $\mathcal{F}/\mathsf{Sem}(\Xi)$ (compare Proposition 4.10 in [2]). This justifies that we can call, in this *atomic case*, the pair $(\mathbb{G}, \eta_{\mathbb{G}})$ a **sketch representation** of $\mathcal{F}$.

Note that the Ξ-structure $(G, \Phi^{\emptyset})$ with $\Phi^{\emptyset}$ a Φ-indexed family of empty sets is trivially freely generated in $\mathsf{Sem}(\Xi)$ by $\mathbb{G} = (G, \emptyset)$.

Second, We Consider Structures Freely Generated Relative to a Subcategory D:

Let D be an arbitrary full subcategory of $\mathsf{Sem}(\Xi)$. A Ξ-structure $\mathcal{F} = (F, \Phi^{\mathcal{F}})$ is freely generated in D by an $\mathtt{Stm}$-sketch $\mathbb{G} = (G, St^{\mathbb{G}})$ if, and only if, $\mathcal{F}$ is an object in D and there is a valid interpretation $(\eta_{\mathbb{G}}, \mathcal{F})$ of $\mathbb{G}$ in $\mathcal{F}$ that is universal relative to D. That is, for all Ξ-structures $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ in D and all valid interpretations $(\iota, \mathcal{U})$ of $\mathbb{G}$ in $\mathcal{U}$ there exists a unique morphism $\iota^* : \mathcal{F} \to \mathcal{U}$ in D such that $\eta_{\mathbb{G}}; \iota^* = \iota$ in Base, i.e., such that ι^* establishes an interpretation morphism $\iota^* : (\eta_{\mathbb{G}}, \mathcal{F}) \to (\iota, \mathcal{U})$ in $\mathtt{Int}(\mathbb{G}) \sqsubseteq \mathtt{Int}(G)$.



A Ξ-structure, freely generated in D by an $\mathtt{Stm}$-sketch $\mathbb{G} = (G, St^{\mathbb{G}})$ is, obviously, uniquely determined "up to isomorphism in D" if it exists. In this case, the universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$ entails that $(\eta_{\mathbb{G}}, \mathcal{F})$ is initial in the subcategory $\mathtt{Int}(\mathbb{G}) \downarrow \mathsf{D} := \Pi_G^{-1}(\mathsf{D})$ of all valid interpretations of $\mathbb{G}$ in Ξ-structures in D. Analogous to the case $\mathsf{D} = \mathsf{Sem}(\Xi)$, we obtain, moreover, an isomorphism between $\mathtt{Int}(\mathbb{G}) \downarrow \mathsf{D}$ and the co-slice category $\mathcal{F}/\mathsf{D}$ if $St^{\mathbb{G}}$ contains only atomic statements.

Third, We Consider Subcategories Described by Logical Means

One logical means to describe subcategories of $\mathsf{Sem}(\Xi)$ are $\mathtt{Stm}$-sketches $(\mathbf{0}, St)$ with St only containing closed formulas, i.e., statements of the form $(\mathbf{0}, Ex, id_{\mathbf{0}})$. As discussed in Remark 21, those sketches can be seen as presentations in the traditional sense of the Theory of Institutions specifying subcategories $\mathsf{Sem}(\Xi, (\mathbf{0}, St))$ of $\mathsf{Sem}(\Xi)$.

As another logical means to describe subcategories of $\mathsf{Sem}(\Xi)$, we will introduce sketch implications in Section 5.2.3 (see Remark 27).

5.2.2. Elementary Diagrams

To establish sketch-based mechanisms to encode structures "syntactically", we have to assume an Institution of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ with $\mathsf{Carr} \sqsubseteq \mathsf{Cxt}$ and $\mathtt{At}(K) \subseteq \mathtt{Stm}(K)$ for all contexts K in Cxt, i.e., $\mathtt{Stm}(K)$ contains all atomic statements in K.

There are two canonical ways to transform a Ξ-structure $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ into an $\mathtt{Stm}$-sketch. The atomic variant $\mathbb{S}^{\mathcal{U}}_{\Phi} = (U, St^{\mathcal{U}}_{\Phi})$ encodes only the semantics of feature symbols and uses therefore only atomic statements:

$$St^{\mathcal{U}}_{\Phi} := \{(\alpha F, F(id_{\alpha F}), \gamma) \mid F \in \Phi, \gamma \in [\![F]\!]^{\mathcal{U}}\} \subseteq \mathtt{At}(U). \tag{12}$$

The full variant $\mathbb{S}^{\mathcal{U}} = (U, St^{\mathcal{U}})$ is available if $\mathtt{XE}(\Xi, X) = \mathtt{FE}(\Xi, X)$ for all $X \in \mathsf{Var}_{Obj}$ and encodes the semantics of all feature expressions:

$$St^{\mathcal{U}} := \{(X, Ex, \gamma) \mid X \in \mathsf{Var}_{Obj},\ Ex \in \mathtt{FE}(\Xi, X),\ \gamma \in [\![Ex]\!]^{\mathcal{U}}_X\} \subseteq \mathtt{Stm}(U). \tag{13}$$

We obviously have $St^{\mathcal{U}}_{\Phi} \subset St^{\mathcal{U}}$. For any statement (X, Ex, γ) in U we obtain, according to (13) and the definition of satisfaction relations in Definition 14,

$$(X, Ex, \gamma) \in St^{\mathcal{U}} \quad \text{iff} \quad \gamma; id_U = \gamma \in [\![Ex]\!]^{\mathcal{U}}_X \quad \text{iff} \quad (id_U, \mathcal{U}) \models_U (X, Ex, \gamma). \tag{14}$$

thus $(id_U, \mathcal{U})$ is a valid interpretation of $\mathbb{S}^{\mathcal{U}}_{\Phi}$ as well as of $\mathbb{S}^{\mathcal{U}}$.

In traditional First-Order Logic, we meet the full variant in the form of elementary diagrams [39]. The difference to our encoding is that the carrier of a first-order structure is not considered as a context. Instead, each element of the carrier is added as a constant to the signature. The encoding of structures as sketches avoids this kind of circularity. The "signature extension trick" works only for first-order signatures with constants symbols and, more critically, it requires that the carriers of first-order structures are sets! It looks like the sketch encoding mechanism is much more flexible and general.

The elementary diagrams in [2] give an abstract account of the signature extension approach but are based on an atomic variant of encoding.

There are no structures at all in [15] only atomic sketches! In [13], we followed Makkai and have not considered structures either. Instead, we worked, directly, with the atomic sketch encodings $\mathbb{S}^{\mathcal{U}}_{\Phi}$ of structures.

To validate, in retrospective, the approaches in [13,15], a noticeable portion of the remaining part of the paper, will be spent to answer the following question:

Question 4: Is there any justification to ignore completely the concept of semantic structure (model)?

By construction, any Ξ-structure $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ in $\mathsf{Sem}(\Xi)$ is freely generated in $\mathsf{Sem}(\Xi)$ by the $\mathtt{Stm}$-sketch $\mathbb{S}^{\mathcal{U}}_{\Phi} = (U, St^{\mathcal{U}}_{\Phi})$ with the universal interpretation $(id_U, \mathcal{U})$. $St^{\mathcal{U}}_{\Phi}$ contains only

atomic statements thus $(\mathbb{S}^{\mathcal{U}}_{\Phi}, id_U)$ becomes a sketch representation of $\mathcal{U}$ in the sense of the last subsection. In particular, there is an isomorphism between $\mathtt{Int}(\mathbb{S}^{\mathcal{U}}_{\Phi})$ and $\mathcal{U}/\mathsf{Sem}(\Xi)$.

The crucial observation is, however, that the assignments $\mathcal{U} \mapsto \mathbb{S}^{\mathcal{U}}_{\Phi}$ define an embedding $\mathtt{Enc} : \mathsf{Str}(\Xi) \to \mathsf{Sk}(\mathtt{Stm})^a$ of $\mathsf{Str}(\Xi)$ into the category $\mathsf{Sk}(\mathtt{Stm})^a$ of all $\mathtt{Stm}$-sketches and all $\mathtt{Stm}$-sketch arrows defined in the next subsection in Definition 20. This embedding establishes, moreover, an isomorphism between $\mathsf{Str}(\Xi)$ and the subcategory $\mathsf{atSk}(\mathtt{Stm})^a_s$ of $\mathsf{Sk}(\mathtt{Stm})^a$ given by all atomic $\mathtt{Stm}$-sketches and all strict $\mathtt{Stm}$-sketch arrows between them. An $\mathtt{Stm}$-sketch $\mathbb{K} = (K, St^{\mathbb{K}})$ is atomic if $St^{\mathbb{K}} \subseteq \mathtt{At}(K)$ (see Remark 14).

The concepts (atomic) $\mathtt{Stm}$-sketch and (strict) $\mathtt{Stm}$-sketch arrow concern only the "structure" of sketches and are completely semantics-independent. That is, the transition along the encoding functor $\mathtt{Enc}$ from the category $\mathsf{Str}(\Xi)$ to the isomorphic category $\mathsf{atSk}(\mathtt{Stm})^a_s$ implements an abstraction from the concept semantic structure (model) to the concept atomic sketch. In case $\mathsf{Sem}(\Xi) = \mathsf{Str}(\Xi)$, this abstraction is exhaustive. In case $\mathsf{Sem}(\Xi)_{Obj} \subsetneq \mathsf{Str}(\Xi)_{Obj}$, however, we need an additional semantics-independent, purely structural characterization identifying exactly all those atomic $\mathtt{Stm}$-sketches $\mathbb{S}^{\mathcal{U}}_{\Phi}$ in $\mathsf{atSk}(\mathtt{Stm})^a_s$ with $\mathcal{U}$ in $\mathsf{Sem}(\Xi)$, to make the abstraction complete (see Remark 29).

A sketch is constituted by a context and a set of statements. The informal term "structure of a sketch" takes into account the context; for each statement, the syntactic structure of the corresponding expression and its "location", i.e., its binding morphism, the set of statements as such and the "distribution" of the statements over the context.

5.2.3. Sketch Arrows and Sketch Implications

We can not only transform Ξ-structures $\mathcal{U}$ into the $\mathtt{Stm}$-sketches $\mathbb{S}^{\mathcal{U}}_{\Phi}$ and $\mathbb{S}^{\mathcal{U}}$. We can even encode the validity of certain classes of "closed formulas" in $\mathcal{U}$ by means of semantics-independent, pure structural closedness properties of $\mathbb{S}^{\mathcal{U}}_{\Phi}$ or $\mathbb{S}^{\mathcal{U}}$, respectively. To see this, we need some preparations.

First, we have to take a step back and consider a very simple, semantics-independent relationship between sketches. To be prepared for Section 6, we define this relationship on the same level of abstraction as Definition 16 (Sketch).

Definition 20 (Sketch Arrow)**.** *Let us be given a category* Ct *and a functor* $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$*. An* ***arrow*** $\varphi : \mathbb{K} \to \mathbb{G}$ *between two* $\mathtt{St}$*-sketches* $\mathbb{K} = (K, St^{\mathbb{K}})$, $\mathbb{G} = (G, St^{\mathbb{G}})$ *is given by a context morphism* $\varphi : K \to G$*.* $\varphi : \mathbb{K} \to \mathbb{G}$ *is called* ***strict*** *if* $St^{\mathbb{G}} \supseteq \mathtt{St}(\varphi)(St^{\mathbb{K}})$*.*

$\mathsf{Sk}(\mathtt{St})^a$ *denotes the category of all* $\mathtt{St}$*-sketches and all* $\mathtt{St}$*-sketch arrows between them. Its subcategory of all* $\mathtt{St}$*-sketches and all strict* $\mathtt{St}$*-sketch arrows is denoted by* $\mathsf{Sk}(\mathtt{St})^a_s$*.*

If $\mathtt{St}$ is clear from the context, we will also use the shorthand notations Sk^a and Sk^a_s instead of $\mathsf{Sk}(\mathtt{St})^a$ and $\mathsf{Sk}(\mathtt{St})^a_s$, respectively. If $K = G$ and $\varphi = id_K$, we will also just write $\mathbb{K} \to \mathbb{G}$ instead of $\varphi : \mathbb{K} \to \mathbb{G}$. We consider the case $\mathsf{Ct} = \mathsf{Cxt}$, $\mathtt{St} = \mathtt{Stm}$ with $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ an Institution of Statements.

Remark 22 (Sketch Morphisms vs. Sketch Arrows)**.** *An* $\mathcal{IS}$*-morphism* $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ *is a* $\mathtt{Stm}$*-sketch arrow* $\varphi : \mathbb{K} \to \mathbb{G}$ *satisfying the semantical* morphism condition $St^{\mathbb{G}} \Vdash_G \mathtt{Stm}(\varphi)(St^{\mathbb{K}})$*. Not every* $\mathtt{Stm}$*-sketch arrow* $\varphi : \mathbb{K} \to \mathbb{G}$ *provides an* $\mathcal{IS}$*-morphism* $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$*, but each* $\mathcal{IS}$*-morphism* $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ *has an* underlying $\mathtt{Stm}$-sketch arrow $\varphi : \mathbb{K} \to \mathbb{G}$*.*

Any strict $\mathtt{Stm}$*-sketch arrow* $\varphi : \mathbb{K} \to \mathbb{G}$ *satisfies, trivially, the morphism condition and provides, in such a way, a strict* $\mathcal{IS}$*-morphism* $\varphi : \mathbb{K} \dashrightarrow \mathbb{G}$ *due to Definition 19.*

There is, however, another semantical condition that is kind of dual to the morphism condition. We call this condition the implication condition and, as a "terminological sleight of hand", we introduce the concept of a "sketch implication", simply indicating that a sketch arrow is intended to be the subject of this dual semantical condition.

Definition 21 (Sketch implication)**.** *An* $\mathcal{IS}$***- implication*** $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is given by two* Stm*-sketches* $\mathbb{P} = (P, St^{\mathbb{P}})$, $\mathbb{C} = (C, St^{\mathbb{C}})$ *and a context morphism* $\varphi : P \to C$.

An $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is called* ***strict*** *if* $St^{\mathbb{C}} \supseteq \mathtt{Stm}(\varphi)(St^{\mathbb{P}})$.

$\mathsf{Sk}(\mathcal{IS})^i$ *denotes the category of all* Stm*-sketches and all* $\mathcal{IS}$*-implications between them. Its subcategory of all* Stm*-sketches and all strict* $\mathcal{IS}$*-implications is denoted by* $\mathsf{Sk}(\mathcal{IS})^i_s$.

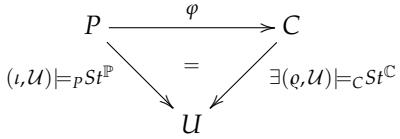
If $\mathcal{IS}$ is clear from the context, we will also use the shorthand notations Sk^i and Sk^i_s instead of $\mathsf{Sk}(\mathcal{IS})^i$ and $\mathsf{Sk}(\mathcal{IS})^i_s$, respectively. If $P = C$ and $\varphi = id_P$, we will also simply write $\mathbb{P} \Rightarrow \mathbb{C}$ instead of $\mathbb{P} \overset{id_P}{\Longrightarrow} \mathbb{C}$.

What we call the implication condition is nothing but the usual condition that an implication is valid if each "solution" of the premise gives rise to a "solution" of the conclusion.

Definition 22 (Validity of Sketch Implications)**.** *Let us be given an* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *between two* Stm*-sketches* $\mathbb{P} = (P, St^{\mathbb{P}})$ *and* $\mathbb{C} = (C, St^{\mathbb{C}})$.

An ***interpretation*** $(\iota, \mathcal{U})$ *of context* P *in a* Ξ*-structure* $\mathcal{U}$ ***satisfies*** $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$, $(\iota, \mathcal{U}) \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *in symbols, if, and only if,* $(\iota, \mathcal{U}) \models_P St^{\mathbb{P}}$ *implies that there exists an interpretation* $(\varrho, \mathcal{U})$ *of context* C *in* $\mathcal{U}$ *with* $\varphi; \varrho = \iota$ *such that* $(\varrho, \mathcal{U}) \models_C St^{\mathbb{C}}$.

$\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is* ***valid in a*** Ξ***- structure*** $\mathcal{U}$, $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *in symbols, if, and only if, we have* $(\iota, \mathcal{U}) \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *for all interpretations* $(\iota, \mathcal{U})$ *of* P *in* $\mathcal{U}$.



$\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is* ***valid (in*** $\mathcal{IS}$***)***, $\models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *in symbols, if, and only if,* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *for all* Ξ*-structures* $\mathcal{U}$ *in* $\mathsf{Sem}(\Xi)$.

Remark 23 (Subcategories of valid Sketch Implications)**.** *For any* Ξ*-structure* $\mathcal{U}$ *and any* Stm*-sketch* $\mathbb{P}$ *we have* $\mathcal{U} \models \mathbb{P} \overset{id_P}{\Rightarrow} \mathbb{P}$. *Moreover,* $\mathcal{U} \models \mathbb{A} \overset{\varphi}{\Rightarrow} \mathbb{B}$ *and* $\mathcal{U} \models \mathbb{B} \overset{\psi}{\Rightarrow} \mathbb{C}$ *implies* $\mathcal{U} \models \mathbb{A} \overset{\varphi;\psi}{\Longrightarrow} \mathbb{C}$. *In such a way, the collection of all* $\mathcal{IS}$*-implications, valid in a* Ξ*-structure* $\mathcal{U}$*, defines a corresponding subcategory of* Sk^i *with the same objects as* Sk^i.

Intersecting all those subcategories for all Ξ*-structures* $\mathcal{U}$ *in* $\mathsf{Sem}(\Xi)$*, we obtain the subcategory* $\mathsf{Sk}^i(\mathsf{Sem}(\Xi))$ *of* Sk^i *given by all* $\mathcal{IS}$*-implications valid in* $\mathcal{IS}$.

Remark 24 (Sketch Implications vs. Sketch Arrows)**.** *An* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is simply another notation for a* Stm*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$. *The only difference is that we allow* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *to be the subject of semantical* implication conditions*, like* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$*, while the corresponding* Stm*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$ *is considered as a pure structural entity without any semantical significance.*

In contrast to $\mathcal{IS}$*-morphisms (compare Remark 22), a strict* Stm*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$ *does not give, trivially, rise to an* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *satisfying semantical* implication conditions.

For any Stm*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$*, we can construct a respective strict* Stm*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}^\varphi$ *with* $\mathbb{C}^\varphi := (C, St^{\mathbb{C}} \cup \mathtt{Stm}(\varphi)(St^{\mathbb{P}}))$. *The satisfaction condition ensures that the corresponding* $\mathcal{IS}$*-implications* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *and* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}^\varphi$ *are semantically equivalent:* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *if, and only if,* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}^\varphi$ *for all* Ξ*-structures* $\mathcal{U}$.

Remark 25 (Sketch Implications vs. Sketch Morphisms)**.** *The concepts* sketch morphism *and* sketch implication *are skewed but kind of dual. Sketch morphisms talk about "reducts of models" while sketch implications state the existence of "model extensions".*

In case $P = C$ and $\varphi = id_P$, a Stm*-sketch arrow $\varphi : \mathbb{P} \to \mathbb{C}$ provides an $\mathcal{IS}$-morphism $id_P : \mathbb{P} \dashrightarrow \mathbb{C}$ if, and only if, $St^{\mathbb{C}} \Vdash_P St^{\mathbb{P}}$ while the validity in $\mathcal{IS}$ of the corresponding $\mathcal{IS}$-implication $\mathbb{P} \Rightarrow \mathbb{C}$ means semantic entailment exactly in the opposite direction $St^{\mathbb{P}} \Vdash_P St^{\mathbb{C}}$!*

Remark 26 (Deduction Rules)**.** *Attention, the exposition in the following remarks and examples, relies implicitly on the observation that sketch arrows can be utilized as* ***deduction rules****. A deduction rule, given by a* Stm*-sketch arrow $\varphi : \mathbb{P} \to \mathbb{C}$, is* ***sound*** *for a certain Institution of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ if, and only if, the respective $\mathcal{IS}$-implication $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ is valid in $\mathcal{IS}$!*

The utilization of sketch arrows as deduction rules is triggered by Definition 23 as well as Proposition 3 and Corollary 2 at the end of this subsection and will be discussed shortly in Remark 32.

Remark 27 (Valid Sketch Implications and Axioms)**.** *There are $\mathcal{IS}$-implications (or, more precisely, $\mathcal{IS}$-implication schemata) that are* ***universal*** *in the sense that they are valid in any Institution of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$, since they reflect the structure and semantics of feature expressions. In particular, the* introduction *and* elimination *rules for logical connectives can be described by those universal sketch implications. In case of* ***conjunction*** *$\wedge$, for example, we do have the two* Stm*-sketches $\mathbb{L} = (X, \{(X, Ex_1 \wedge Ex_2, id_X)\})$ and $\mathbb{R} = (X, \{(X, Ex_1, id_X), (X, Ex_2, id_X)\})$. In addition, the "elimination rule" $\mathbb{L} \Rightarrow \mathbb{R}$ as the "introduction rule" $\mathbb{R} \Rightarrow \mathbb{L}$ are universal sketch implications.*

For existential quantification*, we do also have a kind of* ***modus ponens*** *at hand described by the following universal sketch implication:*

$$(X, \{(X, Ex_1, id_X), (X, Ex_1 \to \exists(\varphi, Y : Ex), id_X)\}) \overset{\varphi}{\Longrightarrow} (Y, \{(Y, Ex_2, id_Y)\}). \tag{15}$$

The validity of other $\mathcal{IS}$-implications may only depend on the chosen base category Base *of an Institution of Statements. As long as* Base *is a presheaf topos, we do have, for example, $\mathcal{IS}$-implications at hand expressing reflexivity, symmetry and transitivity of* equality*, i.e., reflecting the properties of identifications of entities by means of maps (compare the definition of the Ξ_{CT}-expression $[=]$ in Example 29).*

Besides universal $\mathcal{IS}$-implications, we do also have $\mathcal{IS}$-implications that are valid in all Ξ-structures $\mathcal{U}$, and we have chosen to be in $\mathsf{Sem}(\Xi)$. In case $\mathsf{Sem}(\Xi)_{Obj} \subsetneq \mathsf{Str}(\Xi)_{Obj}$, we may be able to axiomatize $\mathsf{Sem}(\Xi)$ in the sense that there is a set SEM of $\mathcal{IS}$-implications such that $\mathsf{Sem}(\Xi) = \mathsf{Str}(\Xi, SEM)$ where $\mathsf{Str}(\Xi, SEM)$ is the full subcategory of $\mathsf{Str}(\Xi)$ given by all those Ξ-structures $\mathcal{U}$ such that $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ for all $\mathcal{IS}$-implications $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ in SEM.

In the same way, we can utilize any set IMP of $\mathcal{IS}$-implications as a set of ***axioms*** *describing the full subcategory $\mathsf{Sem}(\Xi, IMP)$ of $\mathsf{Sem}(\Xi)$ given by all those Ξ-structures $\mathcal{U}$ in $\mathsf{Sem}(\Xi)$ such that $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ for all $\mathcal{IS}$-implications $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ in IMP.*

At the end of Section 5.2.1, we described a mechanism to define subcategories of $\mathsf{Sem}(\Xi)$ by means of axioms in the traditional Hilbert-style, i.e., by Stm*-sketches $(\mathbf{0}, St)$ with St only containing* closed formulas*, i.e., statements of the form $(\mathbf{0}, Ex, id_{\mathbf{0}})$. This mechanism can be integrated in the sketch implication based axiomatization mechanism, in a trivial way, by simply adding to IMP a corresponding* introduction rule *$(\mathbf{0}, \emptyset) \Rightarrow (\mathbf{0}, St)$. For certain classes of closed formulas, there exist more elaborated transformations of Hilbert-style axioms into sketch implication based axioms, as discussed in Section 5.2.4.*

Example 45 (FOL: Sketch implications)**.** *Horn clauses are defined and utilized in PROLOG, in a way that it seems to be appropriate to consider them as sketch implications rather than universally quantified implications. That is, we consider a Horn clause not as a closed formula $(\mathbf{0}, \forall(X : Ex \to Ex'), id_{\mathbf{0}})$ with Ex a finite conjunction of atomic expressions $X \triangleright F_i(\beta_i)$, $\beta_i : \alpha F_i \to X$ with $1 \leq i \leq n$ and an atomic expression $X \triangleright Ex' = F(\beta)$, $\beta : \alpha F \to X$, but rather as the corresponding $\mathcal{IS}$-implication $\mathbb{P} \Rightarrow \mathbb{C}$ with $\mathbb{P} = (X, St^{\mathbb{P}})$, $St^{\mathbb{P}} = \{(\alpha F_i, F_i(id_{\alpha F_i}), \beta_i) \mid 1 \leq i \leq n\}$ and $\mathbb{C} = (X, St^{\mathbb{C}})$, $St^{\mathbb{C}} = \{(\alpha F_i, F(id_{\alpha F}), \beta)\}$.*

Example 46 (ALC: Sketch implications). *A so-called TBox in ALC is a finite set of* terminological axioms, *i.e., of* general concept inclusions $\mathtt{C} \sqsubseteq \mathtt{D}$. *For the way the semantics of* general concept inclusions *is defined in ALC, they correspond, analogous to Horn clauses, rather to sketch implications (than to closed formulas):*

$$(\{p_1\}, \{(\{p_1\}, \mathtt{C}(p_1), id_{\{p_1\}})\}) \Longrightarrow (\{p_1\}, \{(\{p_1\}, \mathtt{D}(p_1), id_{\{p_1\}})\})$$

Example 47 (Category Theory: Sketch implications). *There are, at least, three ways to axiomatize that all vertices do have an identity. First, we can require, due to Remark 18:*

$$\mathcal{U} \models (\mathbf{0}, \forall(X : \exists(in, \alpha_{CT}(\mathtt{id}) : \mathtt{id}(id_{\alpha_{CT}(\mathtt{id})}))), id_{\mathbf{0}})$$

where graph X consists only of a vertex xv and $in : X \to \alpha_{CT}(\mathtt{id})$ is the inclusion of X into $\alpha_{CT}(\mathtt{id})$ (see Examples 14 and 29). As proposed in Remark 27, we can equivalently add the introduction rule:

$$(\mathbf{0}, \emptyset) \Longrightarrow (\mathbf{0}, \{(\mathbf{0}, \forall(X : \exists(in, \alpha_{CT}(\mathtt{id}) : \mathtt{id}(id_{\alpha_{CT}(\mathtt{id})}))), id_{\mathbf{0}})\})$$

to our axioms. According to a general pattern, discussed in the next Section 5.2.4, we can use, instead, the equivalent rule:

$$(X, \emptyset) \Longrightarrow (X, \{(X, \exists(in, \alpha_{CT}(\mathtt{id}) : \mathtt{id}(id_{\alpha_{CT}(\mathtt{id})})), id_X)\}).$$

In turn, this second rule can be composed with a simple variant of the modus ponens rule (15), and we obtain a third equivalent rule:

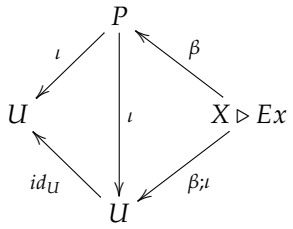
$$(X, \emptyset) \overset{in}{\Longrightarrow} (\alpha_{CT}(\mathtt{id}), \{(\alpha_{CT}(\mathtt{id}), \mathtt{id}(id_{\alpha_{CT}(\mathtt{id})})), id_{\alpha_{CT}(\mathtt{id})})\})$$

Example 48 (RM: Sketch implication). *In DPF, we worked, until now, only with atomic statements and atomic sketch implications called* universal constraint. *In [18,21], the reader can find many examples of those sketch implications expressing properties like: any table should have exactly one primary key, a foreign key has to refer to a primary key, and many, many others. In Remark 28, we will relate the present DPF-terminology to the concepts introduced in this paper.* □

In the remaining part of the subsection, we demonstrate how to encode the validity of sketch implications by a semantics-independent, pure structural closedness property of the sketch encodings $\mathbb{S}^{\mathcal{U}} = (U, St^{\mathcal{U}})$ of Ξ-structures $\mathcal{U}$ as defined in (13).

For any context P any statement (X, Ex, β) in P and any interpretation $(\iota, \mathcal{U})$ of P in a Ξ-structure $\mathcal{U}$, we have $\mathtt{Stm}(\iota)(X, Ex, \beta) = (X, Ex, \beta; \iota)$, due to (2), thus the satisfaction condition and the equivalences in (14) provide the following equivalence of statements:

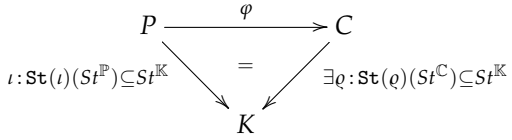
$$(\iota, \mathcal{U}) \models_P (X, Ex, \beta) \quad \text{iff} \quad (id_U, \mathcal{U}) \models_U (X, Ex, \beta; \iota) \quad \text{iff} \quad \mathtt{Stm}(\iota)(X, Ex, \beta) \in St^{\mathcal{U}} \tag{16}$$



To be prepared for Section 6, we define the closedness property on the same level of abstraction as Definition 16 (Sketch) and Definition 20 (Sketch Arrow).

Definition 23 (Closedness). *Let us be given a category* Ct *and a functor* $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$. *A* $\mathtt{St}$*-sketch* $\mathbb{K} = (K, St^{\mathbb{K}})$ *is* ***closed w.r.t. a*** $\mathtt{St}$***- sketch arrow*** $\varphi : \mathbb{P} \to \mathbb{C}$ ***relative to a strict***

`St`- ***sketch arrow*** $\iota : \mathbb{P} \to \mathbb{K}$ *if, and only if, there exists a strict* `St`*-sketch arrow* $\varrho : \mathbb{C} \to \mathbb{K}$ *such that* $\iota = \varphi ; \varrho$.

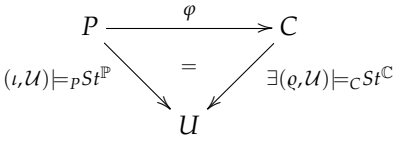


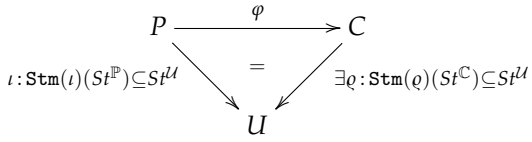
A `St`*-sketch* $\mathbb{K}$ *is* ***closed w.r.t. a*** `St`***- sketch arrow*** $\varphi : \mathbb{P} \to \mathbb{C}$ *if, and only if, it is closed w.r.t.* $\varphi : \mathbb{P} \to \mathbb{C}$ *relative to each strict* `St`*-sketch arrow* $\iota : \mathbb{P} \to \mathbb{K}$.

We consider the case $\mathsf{Ct} = \mathsf{Cxt}$, $\mathtt{St} = \mathtt{Stm}$ with $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ an Institution of Statements. From Definition 22, Definition 23 and Equation (16), we obtain immediately:

Proposition 3 (Validity ≅ Closedness)**.** *For any* `Stm`*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$*, the following two statements are equivalent for any* Ξ*-structure* $\mathcal{U}$*:*

1. *The corresponding* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is valid in* $\mathcal{U}$*, i.e.,* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$.
2. *The* `Stm`*-sketch* $\mathbb{S}^{\mathcal{U}} = (U, St^{\mathcal{U}})$*, defined by (13), is closed w.r.t.* $\varphi : \mathbb{P} \to \mathbb{C}$.





In case of arrows between atomic sketches, we can obviously replace $\mathbb{S}^{\mathcal{U}}$ by $\mathbb{S}^{\mathcal{U}}_{\Phi}$.

Corollary 2 (Validity ≅ Closedness)**.** *For any* `Stm`*-sketch arrow* $\varphi : \mathbb{P} \to \mathbb{C}$ *with* $\mathbb{P}$ *and* $\mathbb{C}$ *atomic, the following two statements are equivalent for any* Ξ*-structure* $\mathcal{U}$*:*

1. *The corresponding* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *is valid in* $\mathcal{U}$*, i.e.,* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$.
2. *The atomic* `Stm`*-sketch* $\mathbb{S}^{\mathcal{U}}_{\Phi} = (U, St^{\mathcal{U}})$*, defined by Equation (13), is closed w.r.t.* $\varphi : \mathbb{P} \to \mathbb{C}$.

Remark 28 (DPF – Answer to Question 3)**.** *In DPF, we worked, until now, only with atomic statements and we have not considered sketch arrows [18,21]. Having now the concept* sketch arrow *explicitly at hand, we can gain a better understanding of the present situation in DPF and are able to answer Question 3 (p. 7).*

The "specification morphisms" in DPF are strict sketch morphisms in the sense of Definition 19. "Specification entailments" in DPF are sketch implications in the sense of Definition 21 but only of the special kind $\mathbb{P} \overset{id_P}{\Longrightarrow} \mathbb{C}$*, i.e., we have, especially,* $P = C$*. The validity of specification entailments is defined analogously to the validity of sketch implications in Definition 22.*

"Universal constraints" in DPF correspond to strict sketch arrows in the sense of Definition 20, and we defined the semantics of universal constraints in accordance with Definition 23. The crucial flaw is that we used, unfortunately and inadequately, the concept specification morphism *to define universal constraints and the closedness property. Effectively, we utilized only the pure structural "strict sketch arrow feature" of DPF specification morphisms for this purpose. However, because of the semantic denotation of the concept* specification morphism*, this was wrong and caused confusion.*

We touched upon the construction of strict sketch arrows $\varphi : \mathbb{P} \to \mathbb{C}^{\varphi}$*, as discussed in Remark 24, but only in the skewed understanding that "each specification entailment gives rise to a universal constraint". Besides this, we have been aware and utilized the observation that "each universal constraint gives rise to a* transformation rule*" (compare Remarks 26 and 32).*

5.2.4. Sketch Implications, Closed Formulas and Makkai's Generalized Sketches

A closer look at the definition of validity of sketch implications in Definition 22 and at the definition of the semantics of feature expressions in Definition 10 makes, straightfor-

wardly, it apparent that the definition of the satisfaction relation in Definition 14 establishes an equivalence between finite sketch implications and universally quantified conditional existence statements (see also Remark 18).

Proposition 4 (Sketch Implications $\cong$ Closed Formulas)**.** *For any Ξ-structure $\mathcal{U}$ and any closed expression $\mathbf{0} \triangleright \forall(X : Ex \to \exists(\varphi, Y : Ex'))$, the following two statements are equivalent:*

1. $\mathcal{U} \models (\mathbf{0}, \forall(X : Ex \to \exists(\varphi, Y : Ex')), id_{\mathbf{0}})$
2. $\mathcal{U} \models (X, \{(X, Ex, id_X)\}) \overset{\varphi}{\Longrightarrow} (Y, \{(Y, Ex', id_Y)\})$

In the case that Ex and Ex' are conjunctions, we can be even more specific.

Corollary 3 (Sketch Implications $\cong$ Closed Formulas)**.** *For any Ξ-structure $\mathcal{U}$ and any closed expression $\mathbf{0} \triangleright \forall(X : Ex \to \exists(\varphi, Y : Ex'))$ with $Ex = Ex_1 \wedge \ldots \wedge Ex_n$ $1 \leq n$ and $Ex' = Ex'_1 \wedge \ldots \wedge Ex'_m$ $1 \leq m$, the following two statements are equivalent:*

1. $\mathcal{U} \models (\mathbf{0}, \forall(X : Ex \to \exists(\varphi, Y : Ex')), id_{\mathbf{0}})$
2. $\mathcal{U} \models (X, \{(X, Ex_1, id_X), \ldots, (X, Ex_n, id_X)\}) \overset{\varphi}{\Longrightarrow} (Y, \{(Y, Ex'_1, id_Y), \ldots, (X, Ex'_m, id_X)\})$

Finally, we can specialize the equivalence to conjunctions of atomic statements.

Corollary 4 (Sketch Implications $\cong$ Closed Formulas: Atomic-case)**.** *For any Ξ-structure $\mathcal{U}$ and any closed expression $\mathbf{0} \triangleright \forall(X : Ex \to \exists(\varphi, Y : Ex'))$ with Ex a finite conjunction of atomic expressions $X \triangleright F_i(\beta_i)$, $\beta_i : \alpha F_i \to X$, $1 \leq i \leq n$ and Ex' a finite conjunction of atomic expressions $Y \triangleright F'_i(\beta'_i)$, $\beta'_i : \alpha F'_i \to Y$, $1 \leq i \leq m$, the following two statements are equivalent:*

1. $\mathcal{U} \models (\mathbf{0}, \forall(X : Ex \to \exists(\varphi, Y : Ex')), id_{\mathbf{0}})$
2. $\mathcal{U} \models (X, \{(\alpha F_i, F_i(id_{\alpha F_i}), \beta_i) \mid 1 \leq i \leq n\}) \overset{\varphi}{\Longrightarrow} (Y, \{(\alpha F'_i, F'_i(id_{\alpha F'_i}), \beta'_i) \mid 1 \leq i \leq m\})$

Now we are sufficiently prepared to give a reasonable answer to Question 4 (p. 39).

Remark 29 (Answer to Question 4)**.** *We discussed in Section 5.2.2 that the assignments $\mathcal{U} \mapsto \mathbb{S}^{\mathcal{U}}_{\Phi}$ establish an isomorphism between $\mathsf{Str}(\Xi)$ and the subcategory $\mathsf{atSk}(\mathtt{Stm})^a_s$ of $\mathsf{Sk}(\mathtt{Stm})^a$ given by all atomic $\mathtt{Stm}$-sketches and all strict $\mathtt{Stm}$-sketch arrows between them.*

A sufficient condition to complete the abstraction from structures to atomic sketches is the existence of a set SEM of atomic (!) $\mathcal{IS}$-implications such that $\mathsf{Sem}(\Xi) = \mathsf{Str}(\Xi, SEM)$ (see Remark 27). The "purely structural characterization of exactly all those atomic $\mathtt{Stm}$-sketches $\mathbb{S}^{\mathcal{U}}_{\Phi}$ in $\mathsf{atSk}(\mathtt{Stm})^a_s$ with $\mathcal{U}$ in $\mathsf{Sem}(\Xi)$", we have been asking for in Section 5.2.2, is then provided by Corollary 2 and is nothing but the closedness of an atomic $\mathtt{Stm}$-sketch w.r.t. all the $\mathtt{Stm}$-sketch arrows underlying the atomic $\mathcal{IS}$-implications in SEM.

If we are only interested in those subcategories $\mathsf{Sem}(\Xi, IMP) = \mathsf{Str}(\Xi, SEM \cup IMP)$ of $\mathsf{Sem}(\Xi)$ which can be axiomatized by a set IMP of atomic (!) $\mathcal{IS}$-implications, we can indeed completely forget about structures and can be content with the "universe of atomic sketches".

This is exactly Makkai's approach in [15]. He does not consider Ξ-structures at all. He relies, instead, on categories $\mathsf{atSk}(\mathtt{Stm})^a_s$ of atomic $\mathtt{Stm}$-sketches and strict $\mathtt{Stm}$-sketch arrows between them. He uses the term sketch entailment *for those* strict sketch arrows *which are utilized for specification purposes. Note that the restriction to* strict sketch arrows *means that he works exclusively with sketch entailments of the form $\varphi : \mathbb{P} \to \mathbb{C}^{\varphi}$ (see Remark 24).*

In the terminology of Institutions of Statements, Makkai's approach can be characterized by the choices $\mathsf{Base} = \mathsf{Var} = \mathsf{Cxt}$ and $\mathtt{Stm}(K) = \mathtt{At}(K)$ for all objects K in Cxt. In particular, he focuses on presheaf topoi, i.e., functor categories $\mathsf{Base} = [\mathsf{C} \to \mathsf{Set}]$, as base categories.

The structure of limit and colimit statements (see Remark 10) is strongly related to the structure of those closed formulas that can be equivalently described by atomic sketch implications (see Corollary 4). We guess that this is one of the underlying reasons that Makkai contents himself with atomic sketches and sketch arrows between them?

We should mention, however, that Makkai uses an additional mechanism to be able to reside in the "universe of atomic sketches". This mechanism (also known in Category Theory as the collage or the cograph of a distributor/profunctor) transforms atomic multi sketches for a presheaf topos $\mathsf{Base} = [\mathsf{C} \to \mathsf{Set}]$ *and a certain footprint* $\Xi = (\Phi, \alpha)$ *on* Base *into plain contexts, i.e., into objects in another presheaf topos* $\mathsf{Base}' = \mathsf{Cxt}' = [\Phi\vec{\alpha}\mathsf{C} \to \mathsf{Set}]$ *with* $\Phi\vec{\alpha}\mathsf{C}$ *a category constructed out of* C *and* Ξ*. We explain and exemplify this construction in Remark 41 in Section 6.*

In cases where atomic sketch implications (and thus the corresponding universally quantified conditional existence statements *in Corollary 3) are not expressive enough to axiomatize the structures* $\mathcal{U}$*, we are interested in, and where we need more expressive first-order statements, to do the job, we can utilize the general first-order sketch constraints, introduced in Section 6, for a pure structural characterization of the respective atomic sketch encodings* $\mathbb{S}^{\mathcal{U}}_{\Phi}$ *(see Corollary 7).*

5.2.5. A Semantic Deduction Theorem

We consider semantic entailment between sketch implications.

Definition 24 (Entailment of Sketch Implications)**.** *A set IMP of* $\mathcal{IS}$*-implications* ***entails an*** $\mathcal{IS}$***- implication*** $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ ***semantically****, IMP* $\Vdash \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$ *in symbols, if, and only if, for all* Ξ*-structures* $\mathcal{U}$ *in* $\mathsf{Sem}(\Xi)$*, it holds that* $\mathcal{U} \models IMP$*, i.e.,* $\mathcal{U} \models \mathbb{K} \overset{\varphi}{\Rightarrow} \mathbb{G}$ *for all* $\mathbb{K} \overset{\varphi}{\Rightarrow} \mathbb{G}$ *in IMP, implies* $\mathcal{U} \models \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$*.*

Any $\mathtt{Stm}$-sketch arrow $\varphi : \mathbb{P} \to \mathbb{C}$ can be factorized, i.e., can be obtained by composing the $\mathtt{Stm}$-sketch arrows $\varphi : \mathbb{P} \to C^{\varphi}$ and $id_C : C^{\varphi} \to \mathbb{C}$, where $C^{\varphi} := (C, \mathtt{Stm}(\varphi)(St^{\mathbb{P}}))$.

Due to Definition 22, for any interpretation $(\iota, \mathcal{U})$ of context P in a Ξ-structure $\mathcal{U}$ the statement $(\iota, \mathcal{U}) \models \mathbb{P} \overset{\varphi}{\Rightarrow} C^{\varphi}$ means nothing but simply the existence of a morphism $\varrho : C \to U$ in Base such that $\varphi; \varrho = \iota$. That is, we have, especially, $IMP \Vdash \mathbb{P} \overset{\varphi}{\Rightarrow} C^{\varphi}$ if, and only if, $\emptyset \Vdash \mathbb{P} \overset{\varphi}{\Rightarrow} C^{\varphi}$ if, and only if, $\models \mathbb{P} \overset{\varphi}{\Rightarrow} C^{\varphi}$. Moreover, this allows for reformulating the validity of sketch implications.

Lemma 1 (Factorization of Sketch Implications)**.** *For any set IMP of* $\mathcal{IS}$*-implications and any* $\mathcal{IS}$*-implication* $\mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$*, the following two statements are equivalent:*

1. $IMP \Vdash \mathbb{P} \overset{\varphi}{\Rightarrow} \mathbb{C}$
2. $\emptyset \Vdash \mathbb{P} \overset{\varphi}{\Rightarrow} C^{\varphi}$ *and* $IMP \Vdash C^{\varphi} \Rightarrow \mathbb{C}$

Lemma 1 means, in practice, that we can restrict ourselves to $\mathcal{IS}$- implications of the form $\mathbb{K} \Rightarrow \mathbb{G}$ to specify (axiomatize) subcategories $\mathsf{Sem}(\Xi, IMP)$ of $\mathsf{Sem}(\Xi)$. By coincidence, we even have a semantic deduction theorem available for those special kinds of sketch implications.

Theorem 2 (Semantic Deduction Theorm)**.** *For any set IMP of* $\mathcal{IS}$*-implications of the form* $\mathbb{K} \Rightarrow \mathbb{G}$ *and any* $\mathcal{IS}$*-implication* $\mathbb{P} \Rightarrow \mathbb{C}$*, the following two statements are equivalent:*

1. $IMP \Vdash \mathbb{P} \Rightarrow \mathbb{C}$
2. *For all* Ξ*-structures* $\mathcal{U}$ *in* $\mathsf{Sem}(\Xi)$ *and all interpretations* $(\iota, \mathcal{U})$ *of context* $P = C$ *in* $\mathcal{U}$*:* $\mathcal{U} \models IMP$ *and* $(\iota, \mathcal{U}) \models_P St^{\mathbb{P}}$ *implies* $(\iota, \mathcal{U}) \models_P St^{\mathbb{C}}$*.*

Theorem 2 guarantees that it is a reasonable idea to describe the deduction of sketch implications, which are semantically entailed by a given set IMP of sketch implications, by means of deduction calculi generating new sketches $(P, St^{\mathbb{C}})$ from a given sketch $(P, St^{\mathbb{P}})$ based on a utilization of the sketch implications in IMP as deduction rules. To deduce all (!) semantically entailed sketch implications, it may be necessary to also include deduction rules related to a set SEM of sketch implications representing the choice of Base, Ξ and $\mathsf{Sem}(\Xi)$, respectively (compare Remarks 27 and 29). This is exactly the approach in [6,7].

In [7], we also presented a deduction calculus which constructs directly sketch implications from a set of given sketch implications. Besides the composition of sketch implications, as mentioned in Remark 23, we could choose parallel composition and instantiation as the other basic constructions for such a deduction calculus:

- *Parallel composition:* $IMP \Vdash (P, St_1) \Rightarrow (P, St_1')$ and $IMP \Vdash (P, St_2) \Rightarrow (P, St_2')$ implies $IMP \Vdash (P, St_1 \cup St_2) \Rightarrow (P, St_1' \cup St_2')$
- *Instantiation:* For any context morphism $\mu : P \to R$: $IMP \Vdash (P, St) \Rightarrow (P, St')$ implies $IMP \Vdash (R, \mathtt{Stm}(\mu)(St)) \Rightarrow (R, \mathtt{Stm}(\mu)(St'))$.

Of course, we could also use, instead, more specialized and sophisticated constructions analogously to *resolution* in PROLOG or *parallel resolution* in [7], for example.

Remark 30 (Resolution in PROLOG)**.** *In Example 45, we argued that Horn clauses in PROLOG should be rather considered as sketch implications than universally quantified implications.*

Concerning resolution*, there is also a discrepancy between the theoretical justification and the actual effect of the resolution procedure in PROLOG. Resolution is explained as a special case of the general principle of "proof by refutation" [40]. Actually, PROLOG computes, however, (in a constructive way!) a Horn clause that is semantically entailed by the Horn clauses and the facts in the given PROLOG program (compare the Semantic Deduction Theorem 2).*

5.3. Sketches of Equations

For an Institutions of Equations $\mathcal{IE} = (\mathsf{Cxt}_{EQ}, \mathtt{Eq}, \mathtt{Int}, \models)$ an $\mathtt{Eq}$-sketch $\mathbb{E} = (X, E)$ is given by a context X in Cxt_{EQ}, i.e., an S-set X, and a set E of Σ-equations in X. A valid interpretation of $\mathbb{E} = (X, E)$ is an interpretation $(\iota, \mathcal{A})$ of context X in a Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ such that $(\iota, \mathcal{A}) \models_X E$, i.e., $(\iota, \mathcal{A}) \models_X (X, t_1 = t_2)$ for all Σ-equations $(X, t_1 = t_2)$ in E according to (9).

Based on these definitions, we can define $\mathcal{IE}$-morphisms, $\mathtt{Eq}$-sketch arrows and $\mathcal{IE}$-implications, respectively, exactly in the same way as we have done it for Institutions of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ in Sections 5.1 and 5.2. Moreover, we have, obviously, for Institutions of Equations also corresponding variants of Definition 22 (Validity of Sketch Implications), Definition 24 (Entailment of Sketch Implications), Lemma 1 (Factorization of Sketch Implications) and Theorem 2 (Semantic Deduction Theorem) available.

Abstract and Universal Algebra have been developed independent of First-Order Logic and conditional Σ-equations are usually not introduced as "universally quantified implications". They are rather described as $\mathcal{IE}$-implications $(Y, Prem) \Rightarrow (Y, Conc)$, in the sense of Definition 21 where *Prem* represents the set of equations in the premise of a conditional Σ-equation and *Conc* the single equation in the conclusion. In particular, the validity of conditional Σ-equations in Σ-algebras $\mathcal{A}$ is defined in perfect accordance with Definition 22 (Validity of Sketch Implications) (compare [6,7,41]). Therefore, we will also use the term **conditional Σ-equation** for $\mathcal{IE}$-implications $(Y, Prem) \Rightarrow (Y, Conc)$ with Y, *Prem* finite and *Conc* a singleton.

Finally, we reached the point where we can give an answer to Question 2 (p. 3): Yes, Theorem 2 is the general Semantic Deduction Theorem, we have been looking for and the equivalence, mentioned in the question, corresponds to the specialization of the general Semantic Deduction Theorem for conditional Σ-equations.

5.3.1. Freely Generated Algebras

The footprints in Institutions of Equations are algebraic signatures $\Sigma = (\Omega, in, out)$ and we have $\mathsf{Str}(\Sigma) = \mathsf{Sem}(\Sigma) := \mathsf{Alg}(\Sigma)$. Conditional Σ-equations are the traditional means to specify subcategories of $\mathsf{Alg}(\Sigma)$. Given a set CE of Conditional Σ-equations, we denote by $\mathsf{Alg}(\Sigma, CE)$ the subcategory of $\mathsf{Alg}(\Sigma)$ given by all those Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \models CE$, i.e., $\mathcal{A} \models \mathbb{P} \Rightarrow \mathbb{C}$ (as defined in Definition 22) for all conditional Σ-equations $\mathbb{P} \Rightarrow \mathbb{C}$ in CE (compare Remark 27).

In case $\mathbb{P} = (Y, \emptyset)$, we may call $\mathbb{P} \Rightarrow \mathbb{C}$ a conditional Σ-equation with an empty premise. Note that there is a simply but crucial conceptual difference between a Σ-equation

$(Y, t_1 = t_2)$ and the corresponding conditional Σ-equation $(Y, \emptyset) \Rightarrow (Y, \{(Y, t_1 = t_2)\})$ with an empty premise. $(Y, t_1 = t_2)$ is just a simple *statement in context* Y while $(Y, \emptyset) \Rightarrow (Y, \{(Y, t_1 = t_2)\})$ is a tool to make statements about Σ-algebras. Being not aware of this difference is often a source of confusion!

First, We Consider Σ-Algebras Freely Generated in $\mathsf{Alg}(\Sigma)$

A Σ-algebra $\mathcal{F} = (F, \Omega^{\mathcal{F}})$ is freely generated by an Eq-sketch $\mathbb{G} = (X, R)$ if, and only if, there is a valid interpretation $(\eta_{\mathbb{G}}, \mathcal{F})$ of $\mathbb{G}$ in $\mathcal{F}$ that is universal relative to $\mathsf{Alg}(\Sigma)$. That is, for all Σ-algebras $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ and all valid interpretations $(\iota, \mathcal{A})$ of $\mathbb{G}$ in $\mathcal{A}$ there exists a unique morphism $\iota^\circ : \mathcal{F} \to \mathcal{A}$ such that $\eta_{\mathbb{G}}; \iota^\circ = \iota$ in Base_{EQ}, i.e., such that ι° establishes an interpretation morphism $\iota^\circ : (\eta_{\mathbb{G}}, \mathcal{F}) \to (\iota, \mathcal{A})$ in $\mathtt{Int}(\mathbb{G}) \sqsubseteq \mathtt{Int}(X)$ (see Section 4.1).

$$\mathtt{Int}(\mathbb{G}) \qquad \begin{array}{ccc} X & \xrightarrow{(\eta_{\mathbb{G}}, \mathcal{F}) \models_X R} & F \\ & \searrow^{(\iota, \mathcal{A}) \models_X R} & \downarrow{\scriptstyle \iota^\circ} \\ & & A \end{array} \qquad \begin{array}{c} \mathcal{F} = (F, \Omega^{\mathcal{F}}) \\ \downarrow{\scriptstyle \iota^\circ} \\ \mathcal{A} = (A, \Omega^{\mathcal{A}}) \end{array} \qquad \mathsf{Alg}(\Sigma)$$

The universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$ entails that $(\eta_{\mathbb{G}}, \mathcal{F})$ is initial in $\mathtt{Int}(\mathbb{G})$, thus the projection functor $\Pi_{\mathbb{G}} : \mathtt{Int}(\mathbb{G}) \to \mathsf{Alg}(\Sigma)$ establishes a functor from $\mathtt{Int}(\mathbb{G})$ into the co-slice category $\mathcal{F}/\mathsf{Alg}(\Sigma)$.

In contrast to Institutions of Statements, we have for arbitrary (!) Eq-sketches $\mathbb{G} = (X, R)$ (and not only for atomic Eq-sketches) that the definition of homomorphisms between Σ-algebras ensures $(\eta_{\mathbb{G}}; \varrho, \mathcal{A}) \models_X R$ for any homomorphism $\varrho : \mathcal{F} \to \mathcal{A}$ in $\mathsf{Alg}(\Sigma)$ thus the assignments $(\varrho : \mathcal{F} \to \mathcal{A}) \mapsto (\eta_{\mathbb{G}}; \varrho, \mathcal{A})$ establish a functor from $\mathcal{F}/\mathsf{Alg}(\Sigma)$ into $\mathtt{Int}(\mathbb{G})$. Due to the universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$, we obtain $(\eta_{\mathbb{G}}; \varrho)^\circ = \varrho$. Together with the equation $\eta_{\mathbb{G}}; \iota^\circ = \iota$, this ensures that the two functors establish an isomorphism between $\mathtt{Int}(\mathbb{G})$ and $\mathcal{F}/\mathsf{Alg}(\Sigma)$ (compare Proposition 4.10 in [2]). This justifies that we can call the pair $(\mathbb{G}, \eta_{\mathbb{G}}) = ((X, R), \eta_{\mathbb{G}})$ a sketch representation of $\mathcal{F}$.

For arbitrary Eq-sketches $\mathbb{G} = (X, R)$, a Σ-algebra $\mathcal{F}$, freely generated by $\mathbb{G}$, exists and is uniquely determined "up to isomorphism". In the introductory Subsection 1.1.1, we used the notation $\mathcal{F}(\Sigma, \emptyset, X, R)$ to denote those freely generated Σ-algebras.

The Σ-algebra, freely generated by $(X, \emptyset)$ is nothing but the Σ-term algebra $\mathcal{T}_\Sigma(X) = \mathcal{F}(\Sigma, \emptyset, X, \emptyset)$ on X and the unique morphism $\iota^\circ : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ is simply the evaluation of terms (see Equation (8)). In general, $\mathcal{F}(\Sigma, \emptyset, X, R)$ can be constructed as a quotient of $\mathcal{T}_\Sigma(X)$.

Second, We Consider Σ-Algebras Freely Generated Relative to a Subcategory $\mathsf{Alg}(\Sigma, CE)$

Let CE be a set of conditional Σ-equations. A Σ-algebra $\mathcal{F} = (F, \Omega^{\mathcal{F}})$ is freely generated in $\mathsf{Alg}(\Sigma, CE)$ by an Eq-sketch $\mathbb{G} = (X, R)$ if, and only if, $\mathcal{F} \models CE$, and there is a valid interpretation $(\eta_{\mathbb{G}}, \mathcal{F})$ of $\mathbb{G}$ in $\mathcal{F}$ that is universal relative to $\mathsf{Alg}(\Sigma, CE)$. That is, for all Σ-algebras $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ in $\mathsf{Alg}(\Sigma, CE)$ and all valid interpretations $(\iota, \mathcal{A})$ of $\mathbb{G}$ in $\mathcal{A}$ there exists a unique morphism $\iota^\circ : \mathcal{F} \to \mathcal{A}$ such that $\eta_{\mathbb{G}}; \iota^\circ = \iota$ in Base_{EQ}, i.e., such that ι° establishes an interpretation morphism $\iota^\circ : (\eta_{\mathbb{G}}, \mathcal{F}) \to (\iota, \mathcal{A})$ in $\mathtt{Int}(\mathbb{G}) \sqsubseteq \mathtt{Int}(X)$.

$$\mathtt{Int}(\mathbb{G}) \downarrow \mathsf{Alg}(\Sigma, CE) \qquad \begin{array}{ccc} X & \xrightarrow{(\eta_{\mathbb{G}}, \mathcal{F}) \models_X R} & F \\ & \searrow^{(\iota, \mathcal{A}) \models_X R} & \downarrow{\scriptstyle \iota^\circ} \\ & & A \end{array} \qquad \begin{array}{c} \mathcal{F} = (F, \Omega^{\mathcal{F}}) \\ \downarrow{\scriptstyle \iota^\circ} \\ \mathcal{A} = (A, \Omega^{\mathcal{A}}) \end{array} \qquad \mathsf{Alg}(\Sigma, CE)$$

In this case, the universal property of $(\eta_{\mathbb{G}}, \mathcal{F})$ entails that $(\eta_{\mathbb{G}}, \mathcal{F})$ is initial in the subcategory $\mathtt{Int}(\mathbb{G}) \downarrow \mathsf{Alg}(\Sigma, CE) = \Pi_G^{-1}(\mathsf{Alg}(\Sigma, CE))$ of $\mathtt{Int}(\mathbb{G})$ given by all valid interpretations of $\mathbb{G}$ in Σ-algebras in $\mathsf{Alg}(\Sigma, CE)$. Moreover, we obtain an isomorphism between $\mathtt{Int}(\mathbb{G}) \downarrow \mathsf{Alg}(\Sigma, CE)$ and the co-slice category $\mathcal{F}/\mathsf{Alg}(\Sigma, CE)$.

For arbitrary sets CE of conditional Σ-equations and arbitrary Eq-sketches $\mathbb{G} = (X, R)$, a Σ-algebra $\mathcal{F}$, freely generated by $\mathbb{G}$ in $\mathsf{Alg}(\Sigma, CE)$, exists and is uniquely determined "up

to isomorphism". In the introductory Section 1.1.1, we used the notation $\mathcal{F}(\Sigma, CE, X, R)$ to denote those freely generated Σ-algebras. $\mathcal{F}(\Sigma, CE, X, R)$ can be constructed as a quotient of $\mathcal{T}_\Sigma(X)$.

In case of groups, CE is a set of conditional Σ-equations with an empty premise, representing the group axioms, and $\mathcal{F}(\Sigma, CE, X, R)$ is called the group freely generated by the set of generators X and the set R of defining relations.

5.3.2. Elementary Diagrams for Algebras

For Institutions of Equations, we have chosen $\mathsf{Cxt}_{EQ} = \mathsf{Carr}_{EQ} = \mathsf{Base}_{EQ} = \mathsf{Set}^S$. An atomic Σ-equation in a context K is a Σ-equation of the form:

$$(K, \omega\langle k_1, \ldots, k_n\rangle = k) \quad \text{with } \omega \in \Omega,\ k_i \in K_{s_i},\ 1 \leq i \leq n \ \text{ and } \ k \in K_{out(\omega)} \tag{17}$$

where $[x_1{:}\,s_1, x_2{:}\,s_2, \ldots, x_n{:}\,s_n]$ is the assumed representation of $in(\omega)$ as a list of variable declarations (see Section 4.2). Note that the usual encoding of n-ary operations by (n + 1)-ary predicates establishes a one-to-one correlation between the corresponding atomic equations and atomic statements, respectively.

By $\mathtt{At}(K)$, we denote the subset of $\mathtt{Eq}(K)$ of all atomic Σ-equation in a context K. The assignments $K \mapsto \mathtt{At}(K)$ extend to a functor $\mathtt{At} : \mathsf{Cxt}_{EQ} \to \mathsf{Set}$.

In full analogy to Institutions of Statements, there are two canonical ways to transform a Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ into an Eq-sketch. The atomic variant $\mathbb{E}^{\mathcal{A}}_\Omega = (A, Eq^{\mathcal{A}}_\Omega)$ encodes only the semantics of the operations in $\Omega^{\mathcal{A}}$:

$$Eq^{\mathcal{A}}_\Omega := \{(A, \omega\langle a_1, \ldots, a_n\rangle = \omega^{\mathcal{A}}(a_1, \ldots, a_n)) \mid \omega \in \Omega, a_i \in A_{s_i},\ 1 \leq i \leq n\} \tag{18}$$

The full variant $\mathbb{E}^{\mathcal{A}} = (A, Eq^{\mathcal{A}})$ encodes the semantics of all terms (derived operations):

$$Eq^{\mathcal{A}} := \{(A, t_1 = t_2) \mid t_1, t_2 \in T_\Sigma(A)_s,\ s \in S,\ t_1^{\mathcal{A}}(id_A) = t_2^{\mathcal{A}}(id_A)\} \subseteq \mathtt{Eq}(A). \tag{19}$$

We have obviously $Eq^{\mathcal{A}}_\Omega \subset Eq^{\mathcal{A}}$ and $(id_{\mathcal{A}}, \mathcal{A})$ is a valid interpretation of $\mathbb{E}^{\mathcal{A}}_\Omega$ as well as of $\mathbb{E}^{\mathcal{A}}$. Any Σ-algebra $\mathcal{A} = (A, \Omega^{\mathcal{A}})$ is freely generated by the Eq-sketch $\mathbb{E}^{\mathcal{A}}_\Omega = (A, Eq^{\mathcal{A}}_\Omega)$ as well as by the Eq-sketch $\mathbb{E}^{\mathcal{A}} = (A, Eq^{\mathcal{A}})$ with the universal interpretation $(id_{\mathcal{A}}, \mathcal{A})$. That is, $(\mathbb{E}^{\mathcal{A}}_\Omega, id_{\mathcal{A}})$ as $(\mathbb{E}^{\mathcal{A}}, id_{\mathcal{A}})$ are sketch representations of $\mathcal{A}$ in the sense of the last subsection.

Conditional Σ-equations are not atomic; thus, we have to rely on the full encodings of Σ-algebras to have a chance to express the validity of conditional Σ-equations by a closedness property analogously to Proposition 3.

Fortunately, the assignments $\mathcal{A} \mapsto \mathbb{E}^{\mathcal{A}}$ define an embedding of $\mathsf{Alg}(\Sigma)$ into the category $\mathsf{Sk}(\mathtt{Eq})^a$ of all Eq-sketches and all Eq-*sketch arrows* transforming each homomorphism between Σ-algebras into a strict Eq-sketch arrow.

5.3.3. Generalized Sketch Arrows and Sketch Implications

To be able to formulate a characterization of the validity of conditional Σ-equations by means of a closedness property, in the sense of Proposition 3, we have to consider more general sketch arrows based on the substitution of variables by terms. First, we extend the category Cxt_{EQ} by Kleisli morphisms.

Definition 25 (Generalized Context Morphisms)**.** *We consider an Institution of Equations* $\mathcal{IE} = (\mathsf{Cxt}_{EQ}, \mathtt{Eq}, \mathtt{Int}, \models)$ *and a signature* $\Sigma = (\Omega, in, out)$ *. A* Σ***- context morphism*** $\varphi : K \to G$ *is given by an S-map* $\varphi : K \to T_\Sigma(G)$*. The composition* $\varphi;\psi : K \to H$ *of two* Σ*-context morphisms* $\varphi : K \to G$*,* $\psi : G \to H$ *is given by the S-map* $\varphi;\psi^* : K \to T_\Sigma(H)$ *where* $\psi^* : T_\Sigma(G) \to T_\Sigma(H)$ *is the usual translation of* Σ*-terms induced by the* ***substitution*** $\psi : G \to T_\Sigma(H)$*.* Cxt^Σ_{EQ} *denotes the category of all contexts and all* Σ*-context morphisms.*

By construction, Cxt_{EQ} is a subcategory of $\mathsf{Cxt}^{\Sigma}_{EQ}$ for any signature Σ. Second, the sentence functor $\mathtt{Eq} : \mathsf{Cxt}_{EQ} \to \mathsf{Set}$ extends to a functor $\mathtt{Eq}^{\Sigma} : \mathsf{Cxt}^{\Sigma}_{EQ} \to \mathsf{Set}$ with:

$$\mathtt{Eq}^{\Sigma}(\varphi)(K, t_1 = t_2) := (G, \varphi^*(t_1) = \varphi^*(t_2)) \tag{20}$$

for all Σ-context morphisms $\varphi : K \to G$, i.e., for all S-maps $\varphi : K \to T_{\Sigma}(G)$, and all Σ-equations $(K, t_1 = t_2)$ in K. Since $id^*_K = id_{T_{\Sigma}(K)}$ and $(\varphi; \psi^*)^* = \varphi^*; \psi^*$ for all S-maps $\varphi\colon K \to T_{\Sigma}(G)$, $\psi\colon G \to T_{\Sigma}(H)$, this defines indeed a functor.

Remark 31 (Generalized Sketch Implications). *We will use, implicitly, strict* $\mathtt{Eq}^{\Sigma}$*-sketch arrows to formulate a characterization of the validity of conditional* Σ*-equations by means of a closedness property in the sense of Proposition 3.*

Besides this, it is very tempting to consider also "generalized sketch implications", defined by $\mathtt{Eq}^{\Sigma}$*-sketch arrows, and to study validity, entailment and factorization for those generalized sketch implications. In particular, it would be interesting to clarify the relation between those generalized sketch implications and the morphisms in the Lawvere theories for partial algebraic specifications we studied in [9].*

For now, we overcome this temptation and postpone the study of generalized sketch implications to a following paper. We will concentrate on conditional Σ*-equations and corresponding constructions and results.*

Before this, we would like to add a short side note: Σ*-terms appear on an "internal level" as constituents of* Σ*-equations and on an "external level" as constituents of generalized context morphisms. Our ongoing studies around* graph algebras *indicate that we will probably need closely related, but different, concepts for these distinct levels if we want to generalize the idea of operations to graphs (and other kinds of presheaves).* □

To avoid headaches, we formulate explicitly the respective instance of Definition 23 for conditional Σ-equations (compare [7]).

Definition 26 (Closedness for Conditional Equations). *An* $\mathtt{Eq}$*-sketch* $\mathbb{E} = (X, E)$ *is* ***closed*** *w.r.t. the underlying* $\mathtt{Eq}$*-sketch arrow* $(Y, Prem) \to (Y, Conc)$ *of a conditional* Σ*-equation* $(Y, Prem) \Rightarrow (Y, Conc)$ *if, and only if, for all* Σ*-context morphisms* $\iota : Y \to X$*, i.e., all substitutions* $\iota : Y \to T_{\Sigma}(X)$*, it holds that* $\iota^*(Prem) \subseteq E$ *implies* $\iota^*(Conc) \subseteq E$*.*

In addition, here is the respective specialized instance of Proposition 3.

Corollary 5 (Validity $\cong$ Closedness for Conditional Equations). *For any* $\mathtt{Eq}$*-sketch arrow* $(Y, Prem) \to (Y, Conc)$*, the following two statements are equivalent for any* Σ*-algebra* $\mathcal{A}$*:*

1. *The corresponding conditional* Σ*-equation* $(Y, Prem) \Rightarrow (Y, Conc)$ *is valid in* $\mathcal{A}$*, i.e.,* $\mathcal{A} \models (Y, Prem) \Rightarrow (Y, Conc)$*.*
2. *The* $\mathtt{Eq}$*-sketch* $\mathbb{E}^{\mathcal{A}} = (A, Eq^{\mathcal{A}})$*, defined by Equation (19), is closed w.r.t. the* $\mathtt{Eq}$*-sketch arrow* $(Y, Prem) \to (Y, Conc)$ *according to Definition 26.*

Remark 32 (Sketch Arrows as Deduction Rules). *The most natural thing to do, if a structure is not closed w.r.t. a certain construction, is to repair this flaw by simply adding the missing parts. Applying this universal "repairing principle" to the closedness property in Definition 26, means nothing but to add new* Σ*-equations to a given set of* Σ*-equations by deploying* $\mathtt{Eq}$*-sketch arrows as deduction rules.*

To apply an $\mathtt{Eq}$*-sketch arrow* $(Y, Prem) \to (Y, Conc)$ *as a deduction rule, we have, first, to find a* ***match*** *of the left-hand side* $(Y, Prem)$ *of the rule in an* $\mathtt{Eq}$*-sketch* $\mathbb{E} = (X, E)$*, i.e., a substitution* $\iota : Y \to T_{\Sigma}(X)$ *such that* $\iota^*(Prem) \subseteq E$*. Second, we apply the rule for this match and generate the* $\mathtt{Eq}$*-sketch* $(X, E \cup \iota^*(Conc))$*. The resulting commutative square becomes a pushout in the category of all strict* $\mathtt{Eq}^{\Sigma}$*-sketch arrows if* $E \cap \iota^*(Conc \setminus Prem) = \emptyset$*.*

$$\begin{array}{ccc} (Y, Prem) & \xrightarrow{id_Y} & (Y, Conc) \\ {\scriptstyle \iota}\downarrow & = & \downarrow{\scriptstyle \tau^*} \\ (X, E) & \xrightarrow{id_X} & (X, E \cup \iota^*(Conc)) \end{array}$$

Remark 33 (Answer to Question 1)**.** *Based on the concepts* sketch, sketch implication, sketch arrow *and the related general definitions and results, we presented so far, we can give a kind of reasonable answer to Question 1 (p.3):*

Each sketch implication has an underlying sketch arrow and, the other way around, each sketch arrow gives rise to a sketch implication. Due to Proposition 3 (Validity ≅ Closedness), the validity of sketch implications in semantic structures can be, moreover, equivalently expressed by a closedness property of sketch encodings of semantic structures w.r.t. sketch arrows.

Each sketch arrow of the form $\mathbb{P} \to \mathbb{C}$ can be utilized as a rule allowing us to deduce new sketches from given sketches (as exemplified in Remark 32). Proposition 3 and Theorem 2 (Semantic Deduction Theorem) ensure that those deductions are sound and that they allow us, moreover, to deduce sketch implications semantically entailed by a given set of sketch implications.

6. Sketch Conditions and Constraints

In the preceding sections, we identified two main motivations to develop concepts and tools, deploying the expressiveness of first-order logic, to describe and reason about the structure of sketches. First, there is the need for those tools to specify the syntactic structure of software models. The second, more general, motivation concerns the structure of sketch encodings of semantic structures. If we use first-order tools to axiomatize the semantic structures we are interested in, it would be good to have corresponding first-order tools to axiomatize and reason about the sketch encodings of those semantic structures.

Software models are usually graph-based structures, thus we should not ignore the concepts and tools, developed in the area of Graph Transformations, to describe and axiomatize the structure of graphs. Therefore, we discuss in this section also four representative first-order based approaches to describe and axiomatize the structure of (different kinds of) graphs [22–25] . We will present a universal and fully first-order mechanism to describe the structure of sketches which unifies and generalizes all these approaches.

6.1. Abstract Sketches

In this section, we consider sketches independent of Institutions of Statements or Institutions of Equations, respectively. That is, we rely on Definition 16 (Sketch) and assume a category Ct of contexts and a functor $\mathtt{St}\colon \mathsf{Ct} \to \mathsf{Set}$ assigning to each $K \in \mathsf{Ct}_{Obj}$ a set $\mathtt{St}(K)$ of all statements in context K. An $\mathtt{St}$-sketch $\mathbb{K} = (K, St^{\mathbb{K}})$ is given by a context K in Ct and a set $St^{\mathbb{K}} \subseteq \mathtt{St}(K)$ of statements in context K. For any statement $st \in \mathtt{St}(K)$, we will denote the image $\mathtt{St}(\varphi)(st) \in \mathtt{St}(G)$ also simply by $\varphi(st)$.

Guided by Definition 23 (Closedness) and Proposition 3 (Validity ≅ Closedness), we focus on the category Sk_s^a of all strict $\mathtt{St}$-sketch arrows according to Definition 20 (Sketch Arrow). Generalizing the constructions and results in [19], one can prove that Sk_s^a has pushouts and pullbacks as long as Ct does.

Proposition 5 (Pushouts)**.** *Let $\mathbb{B} \xleftarrow{\mu} \mathbb{C} \xrightarrow{\varrho} \mathbb{A}$ be a span of strict $\mathtt{St}$-sketch morphisms. If there exists a pushout $B \xrightarrow{\varrho^*} D \xleftarrow{\mu^*} A$ of the span $B \xleftarrow{\mu} C \xrightarrow{\varrho} A$ of morphisms in Ct, then the diagram, below on the left, is a* ***pushout*** *in Sk_s^a, where:*

$$\mathbb{D} := (D, \mu^*(St^{\mathbb{A}}) \cup \varrho^*(St^{\mathbb{B}})) \tag{21}$$

$$\begin{array}{ccc} \mathbb{C} & \xrightarrow{\varrho} & \mathbb{A} \\ {\scriptstyle \mu}\downarrow & PO & \downarrow{\scriptstyle \mu^*} \\ \mathbb{B} & \xrightarrow{\varrho^*} & \mathbb{D} \end{array} \qquad \begin{array}{ccc} \mathbb{D} & \xrightarrow{\mu^*} & \mathbb{A} \\ {\scriptstyle \varrho^*}\downarrow & PB & \downarrow{\scriptstyle \varrho} \\ \mathbb{B} & \xrightarrow{\mu} & \mathbb{C} \end{array}$$

Proposition 6 (Pullbacks). *Let* $\mathbb{B} \xrightarrow{\mu} \mathbb{C} \xleftarrow{\varrho} \mathbb{A}$ *be a cospan of strict* St*-sketch morphisms. If there exists a pullback* $B \xleftarrow{\varrho^*} D \xrightarrow{\mu^*} A$ *of the cospan* $B \xrightarrow{\mu} C \xleftarrow{\varrho} A$ *of morphisms in* Ct*, then the diagram, above on the right, is a* ***pullback*** *in* Sk_s^a *where:*

$$\mathbb{D} := (D, \{st \in \mathtt{St}(D) \mid \mu^*(st) \in St^{\mathbb{A}}, \varrho^*(st) \in St^{\mathbb{B}}\}) \tag{22}$$

Remark 34 (Adhesiveness). *The concept of* Adhesive Category *has been introduced by Lack and Sobociński [42] and is based on the so-called Van-Kampen squares (see [14,20,43]). Adhesive categories are intensively used to present, systematize and generalize concepts, constructions and results in the area of Graph transformations [14]; thus, it seems to be worth including this remark.*

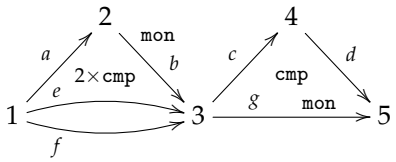
The category Sk_s^a *will be, in general, not adhesive, even if* Ct *is adhesive, since* $St^{\mathbb{D}}$ *in Proposition 5 is not constructed by a pushout in* Set *and in Proposition 6 not by a pullback in* Set *either.*

To repair this deficiency, we can work with "multi sketches" where statements do have their own identity. A ***multi*** St***- sketch*** $\mathbb{K} = (K, I^{\mathbb{K}}, st^{\mathbb{K}})$ *is given by a context* K*, a set* $I^{\mathbb{K}}$ *of identifiers and a map* $st^{\mathbb{K}} : I^{\mathbb{K}} \to \mathtt{St}(K)$*. A* ***strict arrow*** $(\varphi, f) : \mathbb{K} \to \mathbb{G}$ *between two multi* St*-sketches* $\mathbb{K}$ *and* $\mathbb{G}$ *is given by a morphism* $\varphi : K \to G$ *in* Ct *and a map* $f : I^{\mathbb{K}} \to I^{\mathbb{G}}$ *such that* $\varphi(st^{\mathbb{K}}(i)) = st^{\mathbb{G}}(f(i))$ *for all* $i \in I^{\mathbb{K}}$*. Pushouts in the category* mSk_s^a *of multi* St*-sketches and strict arrows can always be constructed by componentwise pushouts of contexts in* Ct *and of sets of identifiers in* Set*, respectively. To ensure that componentwise pullbacks in* Ct *and* Set*, respectively, give us a pullback in* mSk_s^a*, we have to assume, however, that the functor* $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$ *preserves pullbacks.*

This is the case for the sentence functor $\mathtt{Stm} : \mathsf{Cxt} \to \mathsf{Set}$ *in any Institution of Statements as well as the sentence functor* $\mathtt{Eq} : \mathsf{Cxt}_{EQ} \to \mathsf{Set}$ *in any Institution of Equations.*

If St *preserves pullbacks, the monomorphisms in* mSk_s^a *are exactly the componentwise monomorphisms and* mSk_s^a *becomes adhesive if* Ct *is adhesive. Note that any topos is adhesive [44], thus especially the categories* $\mathsf{Cxt}_{EQ} = \mathsf{Set}^S$ *in Institutions of Equations are adhesive.*

Example 49 (Category Theory: Sketches (modified)). *For didactic reasons, we need for this section an example of an atomic sketch. We modify therefore the Category Theory example: We add to* Ξ_{CT} *in Example 14 the feature symbols* mon *with arity* $xv_1 \xrightarrow{xe} xv_2$ *and* fnl *with arity* xv*. Correspondingly, we vary the sample* $\mathtt{Stm}_{CT}$*-sketch* $\mathbb{G} = (G, St^{\mathbb{G}})$ *in Example 39 by dropping the statement* $(xv, fnl, (xv \mapsto 3))$ *and replacing the statements* $(xv_1 \overset{xe}{\to} xv_2, mon, (xe \mapsto b))$*,* $(xv_1 \overset{xe}{\to} xv_2, mon, (xe \mapsto g))$ *by corresponding atomic statements* $\mathtt{mon}(b)$ *and* $\mathtt{mon}(g)$*, respectively.*



Example 50 (GraTra: Sketches). *Traditionally, there is no explicit use of "statements" in the area of Graph Transformations; thus sketches, in our sense, are just plain contexts where different kinds of graphs are chosen as contexts in the different approaches.*

In [23], Cxt *is a category of directed, labeled multi graphs and Ref. [24] restricts* Cxt *to a category of finite directed, labeled multi graphs. In contrast, Ref. [22] works with directed, labeled simple graphs in the sense that parallel edges with the same label are not allowed. Ref. [25] uses as* Cxt *a category* Graph_{TG} *of directed, labeled multi graphs typed over a graph* TG*.*

To a certain extent, we can, however, interpret the transition from graphs to labeled/typed graphs as the utilization of rudimentary forms of "statements", in our sense, where the choice of label alphabets or type graphs TG*, respectively, corresponds to the choice of footprints. The encoding of binary relations by means of labeled edges in [22] makes this analogy apparent. In view of Institutions of Statements, we can reconstruct the concept of graph in [22] in the following way:* Cxt *is the subcategory of* Set *given by all subsets of a "countable universe of nodes* Node*"*

and Var $\sqsubset$ Cxt *has a two-element set* $\{x_1, x_2\} \subset$ Node *as its only object. The footprint* Ξ_R *is given by a "countable universe* Rel*" of predicate symbols with* $\alpha(P) = \{x_1, x_2\}$ *for all* $P \in$ Rel*. An atomic* Ξ_R*-statement* $P(\beta)$ *in context* $K \subseteq$ Node *is, in such a way, given by a* $P \in$ Rel *and a binding* $\beta : \{x_1, x_2\} \to K$ *(see Remark 14 (Atomic Statements)). Relying on the isomorphism between the Cartesian product* $K \times K$ *and the set* $K^{\{x_1,x_2\}}$ *of maps, it is easy to check that the category* Graph *in [22] is isomorphic to the non-adhesive (!) category of all* Ξ_R*-sketches and all strict sketch arrows.*

6.2. First-Order Sketch Conditions and Constraints

Generalizing different variants of graph conditions [14,22–25] as well as universal conditions and negative universal conditions in DPF [18,21], we define general first-order sketch conditions, which are redundant in the sense that we introduce, for example, as well existential as universal quantification and as well a symbol $\boldsymbol{T}$ for "true" as the the empty conjunction $\bigwedge \emptyset$. We define fully fledged first-order conditions and do not restrict ourselves to the traditional approach in Graph Transformations to define tree-like first-order conditions only (even if we see the practical relevance of those tree-like conditions). We define first-order sketch conditions in full analogy to the Definition 8 of first-order feature expressions. We underline, however, that feature expressions are "finitary syntactic entities" while sketch conditions have rather the flavor of sets of structural requirements!

Definition 27 (Sketch conditions: Syntax)**.** *For a category* Ct *and a functor* $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$*, we define inductively and in parallel a family* $\mathtt{ST}(K)$ *of sets of* ***first-order*** $\mathtt{St}$***- sketch conditions in context*** *K,* $c \in \mathtt{ST}(K)$ *or* $K \blacktriangleright c$ *in symbols, where K varies over all objects in* Ct*:*

1. *Statements:* $\mathtt{St}(K) \subset \mathtt{ST}(K)$ *for any context K.*
2. *True:* $K \blacktriangleright \boldsymbol{T}$ *for any context K.*
3. *False:* $K \blacktriangleright \boldsymbol{F}$ *for any context K.*
4. *Conjunction:* $K \blacktriangleright \bigwedge C$ *for any set* $C \subset \mathtt{ST}(K)$ *of conditions in K.*
5. *Disjunction:* $K \blacktriangleright \bigvee C$ *for any set* $C \subset \mathtt{ST}(K)$ *of conditions in K.*
6. *Implication:* $K \blacktriangleright (c_1 \to c_2)$ *for any conditions* $K \blacktriangleright c_1$ *and* $K \blacktriangleright c_2$*.*
7. *Negation:* $K \blacktriangleright \neg c$ *for any condition* $K \blacktriangleright c$*.*
8. *Quantification:* $K \blacktriangleright \exists(\varphi, M : c)$ *and* $K \blacktriangleright \forall(\varphi, M : c)$ *for any condition* $M \blacktriangleright c$ *and any morphism* $\varphi : K \to M$ *in* Ct *that is not an isomorphism.*

Remark 35 (Sketch conditions: Syntax)**.** *Non-monic morphisms* $\varphi : K \to M$ *are also used in [22–24] to express identifications.*

For sketch conditions, we apply the same notational conventions as described in Remark 6 for feature expressions.

If $\mathbf{0}$ *is an initial object in* Ct*, we call* $\mathbf{0} \blacktriangleright c$ *a* ***closed*** $\mathtt{St}$***- sketch condition****.*

Remark 36 (GraTra: Conditions)**.** *If we drop in Definition 27 the "Implication" rule, we would obtain tree-like conditions analogously to the conditions in [23–25], where the tree structure is established by the context morphisms in the "Quantification" rule and the choice of the sets C in the "Conjunction" and/or "Disjunction" rule, respectively.*

To cover also the tree-like conditions in [22], we have, in addition, to replace the "Quantification" rule by a rule like:

> *Guarded quantification:* $K \blacktriangleright (c_1 \to \boldsymbol{Q}(\varphi, M : c_2))$ *for* $Q \in \{\exists, \forall\}$*, any* quantifier free *condition* $K \blacktriangleright c_1$*, any condition* $M \blacktriangleright c_2$ *and any morphism* $\varphi : K \to M$ *in* Ct*.*

Those tree-like conditions can be seen as a generalizing modification of the Q(uantifier)-trees *of the* language of diagrams *in [28].*

In [23,25], only existential quantification $\exists(\varphi, M : c)$ *is used and* $\forall(\varphi, M : c)$ *is encoded by* $\neg\exists(\varphi, M : \neg c)$*. In [24], the symbols "*$\exists$*" and "*$\forall$*" are used in a bit unconventional, but consistent, way: In view of Definition 27, the symbol "*$\exists$*" in [24] combines "disjunction and existential quantification" while "*$\forall$*" combines conjunction and universal quantification. The conditions in [24] correspond to sketch conditions that can be generated by a single rule like:*

$$\bigvee\{\exists(\varphi_i, M_i : c_i) \mid i \in I\}, \bigwedge\{\forall(\varphi_i, M_i : c_i) \mid i \in I\} \in \mathtt{ST}(K)$$

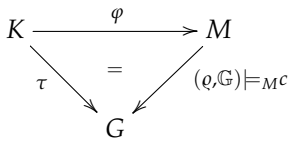
for any family $\{\varphi_i : K \to M_i \mid i \in I\}$ *of context morphisms and any conditions* $c_i \in \mathtt{ST}(M_i)$, $i \in I$. ***T*** *is encoded in [24] by the empty conjunction* $\bigwedge\emptyset$ *and* ***F*** *by the empty disjunction* $\bigvee\emptyset$*, respectively.*

Generalizing the traditional approaches [14,22–25] to define a satisfaction relation between graph morphisms and graph conditions, we can define a satisfaction relation between context morphisms and sketch conditions.

More precisely, we consider interpretations $(\tau : K \to G, \mathbb{G})$ of contexts K in $\mathtt{St}$-sketches $\mathbb{G} = (G, St^{\mathbb{G}})$ and define valid interpretations of $\mathtt{St}$-sketch conditions in K.

Definition 28 (Sketch conditions: Satisfaction)**.** *We define inductively and in parallel a family* $\models_K$ *of satisfaction relations between interpretations* $(\tau, \mathbb{G})$ *of contexts* K *in* $\mathtt{St}$*-sketches* $\mathbb{G} = (G, St^{\mathbb{G}})$ *and* $\mathtt{St}$*-sketch conditions* $c \in \mathtt{ST}(K)$ *on* K*:*

1. *Statement: For all* $st \in \mathtt{St}(K) \subset \mathtt{ST}(K)$: $(\tau, \mathbb{G}) \models_K st$ *iff* $\mathtt{St}(\tau)(st) \in St^{\mathbb{G}}$.
2. *True:* $(\tau, \mathbb{G}) \models_K \boldsymbol{T}$
3. *False:* $(\tau, \mathbb{G}) \not\models_K \boldsymbol{F}$
4. *Conjunction:* $(\tau, \mathbb{G}) \models_K \bigwedge C$ *iff* $(\tau, \mathbb{G}) \models_K c$ *for every* $c \in C$.
5. *Disjunction:* $(\tau, \mathbb{G}) \models_K \bigvee C$ *iff* $(\tau, \mathbb{G}) \models_K c$ *for some* $c \in C$.
6. *Implication:* $(\tau, \mathbb{G}) \models_K (c_1 \to c_2)$ *iff* $(\tau, \mathbb{G}) \models_K c_1$ *implies* $(\tau, \mathbb{G}) \models_K c_2$
7. *Negation:* $(\tau, \mathbb{G}) \models_K \neg c$ *iff* $(\tau, \mathbb{G}) \not\models_K c$.
8. *Existential quantification:* $(\tau, \mathbb{G}) \models_K \exists(\varphi, M : c)$ *iff there exists a* $\varrho : Y \to G$ *with* $\varphi; \varrho = \tau$ *and* $(\varrho, \mathbb{G}) \models_M c$



Universal quantification: $(\tau, \mathbb{G}) \models_K \forall(\varphi, M : c)$ *iff for all* $\varrho : Y \to G$ *with* $\varphi; \varrho = \tau$ *we have* $(\varrho, \mathbb{G}) \models_M c$

The satisfaction of graph/sketch conditions by a graph/context morphism is a powerful and practical useful tool to control the application of transformation rules. This is extensively demonstrated and validated in the Graph Transformation literature as in [14,22–25], for example. In DPF, we used until now only non-nested negative application conditions to control the application of non-deleting model transformation rules [18,21]. The paper paves the way for utilizing arbitrary first-order conditions to control model transformations in DPF. In this paper, we will, however, not explore this promising direction of applying first-order sketch conditions. We rather concentrate on two other aspects of diagrammatic modeling techniques – namely "syntactic structure" of models and "deducing information from and reason about models" in a diagrammatic manner.

Developing and applying DPF, we realized that typing mechanisms are not powerful enough to formalize all relevant restrictions concerning the syntactic structure of models. To overcome this deficiency, we introduced "universal constraints" and "negative universal constraints" [18,21] analogous to the non-nested graph constraints in [14].

Fortunately, sketch conditions and their satisfaction, as defined in Definition 28, now give us also more powerful general first-order sketch constraints at hand to describe the syntactic structure of models. The simple, but crucial, observation is that an assertion $(\tau, \mathbb{G}) \models_K c$ can be interpreted as well as an assertion concerning the structure of $\mathbb{G}$.

Definition 29 (Sketch constraints)**.** *An* $\mathtt{St}$***- sketch*** *K****- constraint*** (c, τ) *on context* G *is given by a context* K*, a sketch condition* $K \blacktriangleright c$ *in context* K *and a context morphism* $\tau : K \to G$.

An St*-sketch* $\mathbb{G} = (G, St^{\mathbb{G}})$ ***satisfies*** *the K-constraint* (c, τ), $\mathbb{G} \models_K (c, \tau)$ *in symbols, if, and only if,* $(\tau, \mathbb{G}) \models_K c$.

Remark 37 (Attached Statements)**.** *Only at this point and a few days before the paper deadline, we realized that it may be beneficial to apply the "reinterpretation principle" in Definition 29 also to structures. That is, for any* Ξ*-structure* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$*, context K, morphism* $\iota : K \to U$ *and statement* (X, Ex, γ) *in K we can define:*

$$\mathcal{U} \models_K ((X, Ex, \gamma), \iota) \quad \textit{iff} \quad (\iota, \mathcal{U}) \models_K (X, Ex, \gamma) \tag{23}$$

and may call the pair $((X, Ex, \gamma), \iota)$ *a statement attached to* $\mathcal{U}$ *or a statement about* $\mathcal{U}$*.*

To realize this idea, would, however, require a major revision of the paper.

If the sketch condition c does not contain any statements, as it usually the case in the area of Graph Transformations (compare Example 50), $\mathbb{G} \models_K (c, \tau)$ is just an assertion about the structure of the context G. In all other cases, $\mathbb{G} \models_K (c, \tau)$ tells us also something about the presence or non-presence of statements as well as the relations between the statements in $\mathbb{G}$.

Due to rule "Statement", all statements reappear as conditions. The following simple corollary illustrates that the requirement for strict sketch arrows to preserve statements "on the nose" encodes a structural constraint on the target.

Corollary 6 (Strict Sketch Arrow vs. Sketch Constraint)**.** *A context morphism* $\varphi : K \to G$ *constitutes a strict* St*-sketch arrow* $\varphi : \mathbb{K} \to \mathbb{G}$ *between two* St*-sketches* $\mathbb{K} = (K, St^{\mathbb{K}})$ *and* $\mathbb{G} = (G, St^{\mathbb{G}})$ *if, and only if,* $\mathbb{G} \models_K (\bigwedge St^{\mathbb{K}}, \varphi)$.

Remark 38 (General constraints)**.** *A K-constraint* (c, τ) *is, in general, only a* ***local constraint****, in the sense that it constrains the structure of* $\mathbb{G}$ *"around the image" of K w.r.t.* τ*. Thus, in case* $K = G$ *and* $\tau = id_G$*,* (c, id_G) *is an assertion about the structure of* $\mathbb{G}$ *as such.*

If Ct *has an initial object* $\mathbf{0}$*, any closed condition* $\mathbf{0} \blacktriangleright c$ *gives rise to a sketch constraint* $(c, !_G)$ *with* $!_G : \mathbf{0} \to G$ *the initial morphism into G.* $(c, !_G)$ *is a* ***general constraint****, in the sense that the statement* $\mathbb{G} \models_{\mathbf{0}} (c, !_G)$ *can be seen as a characterization of the overall structure of* $\mathbb{G}$*. In the Graph Transformation literature, only general constraints have been considered [23,25].*

6.3. Statements and Sketch Constraints

In this subsection, we outline that first-order sketch constraints give us indeed the means at hand to express the validity of statements in semantic structures, in an equivalent way, by structural properties of sketch encodings of those semantic structures (see Remark 29). In particular, we are interested to extend Makkai's approach and to encode the validity of arbitrary first-order statements in semantic structures by structural properties of atomic sketch encodings.

Thus, we go back to the setting in Section 5.2.2 and assume an Institution of Statements $\mathcal{IS} = (\mathsf{Cxt}, \mathtt{Stm}, \mathtt{Int}, \models)$ with $\mathsf{Carr} \sqsubseteq \mathsf{Cxt}$, $\mathtt{XE}(\Xi) = \mathtt{FE}(\Xi)$ and thus $\mathtt{At}(K) \subset \mathtt{Stm}(K)$ for all contexts K in Cxt, i.e., $\mathtt{Stm}(K)$ contains all *atomic statements* in K.

We consider the instances of Definition 27 and Definition 28, respectively, for the category Cxt of contexts and the functor $\mathtt{At} : \mathsf{Cxt} \to \mathsf{Set}$ assigning to each context K the set $\mathtt{At}(K)$ of all atomic statements in K as described in Remark 14.

Definition 27 of the syntax of sketch conditions follows exactly the same pattern as Definition 8 of the syntax of feature expression; thus, it should be possible to translate, for any context K, the statements in K into $\mathtt{At}$-sketch conditions on K. This is indeed possible! However, to be able to translate *quantifications*, we have to assume that Cxt has pushouts (compare Appendix A).

Definition 30 (From Statements to Sketch conditions)**.** *We assume that* Cxt *has pushouts. For an arbitrary but fixed choice of pushouts in* Cxt *we construct inductively and in parallel a*

family of maps $tr_K : \mathtt{Stm}(K) \to \mathtt{AT}(K)$*, where* K *varies over all objects in* Ct*: For arbitry variable declarations* X *and arbitrary binding morphism* $\gamma\colon X \to K$*, we define*

1. *Atomic expr.:* $tr_K(X, F(\beta), \gamma) := F(\beta;\gamma) = (\alpha F, F(id_{\alpha F}), \beta;\gamma) \in \mathtt{At}(K) \subset \mathtt{AT}(K)$
2. *Everything:* $tr_K(X, \top, \gamma) := \boldsymbol{T} \in \mathtt{AT}(K)$
3. *Void:* $tr_K(X, \bot, \gamma) := \boldsymbol{F} \in \mathtt{AT}(K)$
4. *Conjunction:* $tr_K(X, (Ex_1 \wedge Ex_2), \gamma) := \bigwedge\{tr_K(X, Ex_1, \gamma), tr_K(X, Ex_2, \gamma)\} \in \mathtt{AT}(K)$
5. *Disjunction:* $tr_K(X, (Ex_1 \vee Ex_2), \gamma) := \bigvee\{tr_K(X, Ex_1, \gamma), tr_K(X, Ex_2, \gamma)\} \in \mathtt{AT}(K)$
6. *Implication:* $tr_K(X, (Ex_1 \to Ex_2), \gamma) := (tr_K(X, Ex_1, \gamma) \to tr_K(X, Ex_2, \gamma)) \in \mathtt{AT}(K)$
7. *Negation:* $tr_K(X, \neg Ex, \gamma) := \neg\, tr_K(X, Ex, \gamma) \in \mathtt{AT}(K)$
8. *Quantification:* $tr_K(X, Q(\varphi, Y : Ex), \gamma) := Q(\varphi^*, K^\varphi_\gamma : tr_{K^\varphi_\gamma}(Ex)) \in \mathtt{AT}(K)$

   for $Q \in \{\exists, \forall\}$ *where* $K \xrightarrow{\varphi^*} K^\varphi_\gamma \xleftarrow{\gamma^*} Y$ *is the chosen pushout of* $K \xleftarrow{\gamma} X \xrightarrow{\varphi} Y$.

Remark 39 (Translation of Feature Expressions)**.** *Every feature expression* $X \triangleright Ex$ *reappears as the statement* (X, Ex, id_X)*, thus we can consider* $X \blacktriangleright tr_X(X, Ex, id_X)$ *as the translation of the feature expression* $X \triangleright Ex$ *into a* $\mathtt{At}$*-sketch condition.*

Besides syntax, also Definition 10 of the semantics of feature expressions (and thus Definition 14 of satisfaction of statements) and Definition 28 of satisfaction of sketch conditions (and thus Definition 29 of satisfaction of sketch constraints) follow exactly the same pattern. This enables us to prove straightforwardly that the family of translation maps $tr_K : \mathtt{Stm}(K) \to \mathtt{AT}(K)$ establishes an equivalence between first-order statements and first-order $\mathtt{At}$-sketch conditions. Note that the proposal in Remark 37 would make the statement in the following proposition even more catchy.

Proposition 7 (Statements $\cong$ Sketch Constraints)**.** *For any* Ξ*-structure* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$*, context* K*, morphism* $\iota\colon K \to U$ *and statement* (X, Ex, γ) *in* K *we have:*

$$(\iota, \mathcal{U}) \models_K (X, Ex, \gamma) \quad \textit{iff} \quad \mathbb{S}^{\mathcal{U}}_\Phi \models_K (tr_K(X, Ex, \gamma), \iota),$$

where $\mathbb{S}^{\mathcal{U}}_\Phi = (U, St^{\mathcal{U}}_\Phi)$ *is the atomic sketch encoding of structure* $\mathcal{U}$ *as defined by (12).*

Instantiating this equivalence for the identity on U gives us exactly what we have been looking for, namely that the atomic sketch encoding of structures in an Institution of Statements encodes likewise all properties of structures that can be expressed by first-order statements and formulas.

Corollary 7 (Statements $\cong$ Sketch Constraints)**.** *For any* Ξ*-structure* $\mathcal{U} = (U, \Phi^{\mathcal{U}})$ *and any statement* (X, Ex, γ) *in* U *we have:*

$$(id_U, \mathcal{U}) \models_U (X, Ex, \gamma) \quad \textit{iff} \quad \mathbb{S}^{\mathcal{U}}_\Phi \models_K (tr_K(X, Ex, \gamma), id_U),$$

where $\mathbb{S}^{\mathcal{U}}_\Phi = (U, St^{\mathcal{U}}_\Phi)$ *is the atomic sketch encoding of structure* $\mathcal{U}$ *as defined by (12).*

The case $X = \mathbf{0}$, and thus $\gamma = !_U$, corresponds to closed formulas and, due to Remark 18 (Validity of Closed Formulas), Corollary 7 ensures that we can detect all closed formulas that are valid in $\mathcal{U}$, by inspecting the atomic sketch encoding $\mathbb{S}^{\mathcal{U}}_\Phi$.

Proposition 7 and Corollary 7 are very good news for DPF and any other diagrammatic approach to Software Engineering. They ensure that we can describe both structure and constraints in the same diagrammatic, modelcentric format. There is, in principle, no need to combine diagrammatic models with dissimilar descriptions, like OCL code, for example, even if it comes to first-order properties. We can reason about and deal with a real system at a higher level of abstraction within one and the same diagrammatic paradigm!

Example 51 (CT: Sketch constraints)**.** *Relying on Definition 30 and Remark 39, we can translate all the sample* Ξ_{CT}*-expressions in Example 29 into corresponding sketch conditions. Concerning the*

visual representation, there is no essential difference between a Ξ_{CT}-expression and the corresponding sketch condition: We replace $\triangleright$ by $\blacktriangleright$ and $\top$ by $\boldsymbol{T}$. We rewrite $(\_ \wedge \_)$ to $\bigwedge\{\_,\_\}$ and so on.

The Ξ_{CT}-expressions lec in Example 29 is transformed into the sketch condition

$$\overline{lec} = \quad xv_2 \quad \blacktriangleright \exists(\quad xv_2 \quad : \quad xv_2 \quad)$$

$$xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2}$$

$$xv_1 \qquad xv_3 \qquad xv_1 \xrightarrow{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{cmp}]{xe_3} xv_3$$

and the Ξ_{CT}-expressions gec, representing the property composition is always defined, *is transformed into the sketch condition:*

$$\overline{gec} = \mathbf{0} \blacktriangleright \forall(\quad xv_2 \quad : \exists(\quad xv_2 \quad : \quad xv_2 \quad))$$

$$xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2}$$

$$xv_1 \qquad xv_3 \qquad xv_1 \xrightarrow{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{cmp}]{xe_3} xv_3$$

For the sample sketch $\mathbb{G} = (G, St^{\mathbb{G}})$ in Example 49, we do have $\mathbb{G} \models (\overline{lec}, \tau_1)$, with τ_1 given by the assignments $xe_1 \mapsto a, xe_2 \mapsto b$, but $\mathbb{G} \not\models (\overline{lec}, \tau_2)$, with τ_2 given by $xe_1 \mapsto b, xe_2 \mapsto c$, thus $\mathbb{G} \not\models (\overline{gec}, !_G)$. General constraints imposing uniqueness of composition, *independent of the existence of composition, can be formulated by the closed condition $\overline{guc}$:*

$$\mathbf{0} \blacktriangleright \forall(\quad xv_2 \quad : (\bigwedge\{ \quad xv_2 \quad , \quad xv_2 \quad \} \to \exists(\varphi, \quad xv_2 \quad : \boldsymbol{T} \;)))$$

$$xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2}$$

$$xv_1 \underset{xe_4}{\overset{xe_3}{\rightrightarrows}} xv_3 \qquad xv_1 \xrightarrow[\texttt{cmp}]{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{cmp}]{xe_4} xv_3 \qquad xv_1 \xrightarrow{e} xv_3$$

φ simply maps xe_3 and xe_4 to xe. $\mathbb{G}$ does not satisfy the constraint $(\overline{guc}, !_G)$ but would satisfy it if we delete the edge "f", for example. The remaining requirements – existence and uniqueness of identities, identity laws and associativity law – can be expressed analogously.

Besides formalizing the "laws of a category", we can also take advantage of our knowledge about the properties of the features in Ξ_{CT} – or to put it the other way around: We can formulate requirements that any intended semantics of the feature symbols in Ξ_{CT} has to comply with. For example, we can require that, for a final object, all outgoing morphisms are monic:

$$ct_1 := \mathbf{0} \blacktriangleright \forall(xv : (xv^{\texttt{fnl}} \longrightarrow \forall(xv \xrightarrow{xe} xv_1 : \; xv \xrightarrow[\texttt{mon}]{e} xv_1 \;)))$$

We can require that monomorphisms are closed under composition:

$$ct_2 := \mathbf{0} \blacktriangleright \forall(\quad xv_2 \quad : (\bigwedge\{ \quad xv_2 \quad \} \longrightarrow \quad xv_2 \quad))$$

$$xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2}$$

$$xv_1 \xrightarrow{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{cmp}]{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{mon}]{xe_3} xv_3$$

Note that we use $\bigwedge\{\cdots\}$ because the single triangle between the curly brackets visualizes, actually, three atomic Ξ_{CT}-statements! We can also express our knowledge concerning the decomposition of monomorphisms:

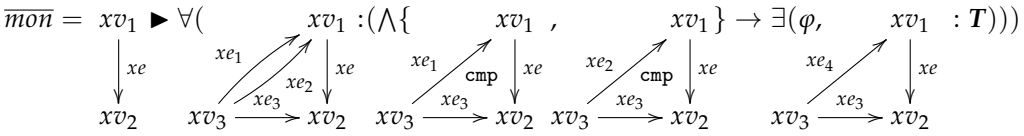
$$ct_3 := \mathbf{0} \blacktriangleright \forall(\quad xv_2 \quad : (\bigwedge\{ \quad xv_2 \quad \} \longrightarrow \quad xv_2 \quad))$$

$$xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2} \qquad xe_1 \uparrow \quad \searrow^{xe_2}$$

$$xv_1 \xrightarrow{xe_3} xv_3 \qquad xv_1 \xrightarrow[\texttt{mon}]{xe_3} xv_3 \qquad xv_1 \xrightarrow{xe_3} xv_3$$

$\mathbb{G} \models (ct_2, !_G)$ simply because there is no match in G of the triangular context in ct_2 satisfying the premise of the implication in ct_2. In contrast, $\mathbb{G} \not\models (ct_3, !_G)$ with the only counterexample given by the assignments $xe_1 \mapsto c, xe_2 \mapsto d, xe_3 \mapsto g$.

To be prepared for discussions, later in this section, we consider also the sketch condition $\overline{mon}$ defining the concept monomorphism *and obtained by transforming the Ξ_{CT}-expressions mon in Example 29:*



where φ maps xe_1 and xe_2 to xe_4.

6.4. Sketch Arrows, Constraints, Deduction, Meta-Modeling

In this subsection, we present vital observations, insights, concepts and ideas to establish a basis for the future further development of the "logic dimension" of Institutions of Statements and, especially of DPF, based on the new concepts and results presented in this paper.

Constraints in DPF at Present

Following [15] and analogous to [14], we use in DPF until now, instead of sketch constraints in the sense of Definition 29, only plain sketch arrows $\varphi : \mathbb{L} \to \mathbb{R}$ and call them (positive) universal constraints or negative universal constraints, respectively [18,21]. We define the satisfaction of universal constraints in DPF by means of the closedness property in Definition 23. That is, a sketch $\mathbb{G}$ satisfies the "universal constraint" $\varphi : \mathbb{L} \to \mathbb{R}$ if, and only if, for any strict sketch arrow $\tau : \mathbb{L} \to \mathbb{G}$ there is a strict sketch arrow $\varrho : \mathbb{R} \to \mathbb{G}$ such that $\varphi; \varrho = \tau$.

By Proposition 7, we can transfer many findings in Sections 5.2.3 and 5.2.4 into the sketch constraints setting. Corollary 6 and Definition 28 ensure that the satisfaction of a universal constraint $\varphi : \mathbb{L} \to \mathbb{R}$ in a sketch $\mathbb{G}$ can be equivalently expressed by the assertion that $\mathbb{G}$ satisfies the general constraint $(uc, !_G)$ with (compare Corollary 3):

$$gc := \mathbf{0} \blacktriangleright \forall(L : (\bigwedge St^{\mathbb{L}} \to \exists(\varphi, R : \bigwedge St^{\mathbb{R}}))).$$

Be aware that the identifier φ in gc does not refer to the sketch arrow $\varphi : \mathbb{L} \to \mathbb{R}$ but to the underlying context morphism $\varphi : L \to R$. Note further that we can replace $St^{\mathbb{R}}$ by $(St^{\mathbb{R}} \setminus \varphi(St^{\mathbb{L}}))$ without losing the equivalence!

Furthermore, we say that a sketch $\mathbb{G}$ satisfies the "negative universal constraint" $\varphi : \mathbb{L} \to \mathbb{R}$ if, and only if, for any strict sketch arrow $\tau : \mathbb{L} \to \mathbb{G}$, there does not exist a strict sketch arrow $\varrho : \mathbb{R} \to \mathbb{G}$ such that $\varphi; \varrho = \tau$. This requirement is equivalent to the statement that $\mathbb{G}$ satisfies the general constraint $(ngc, !_G)$ with:

$$ngc := \mathbf{0} \blacktriangleright \forall(L : (\bigwedge St^{\mathbb{L}} \to \neg\exists(\varphi, R : \bigwedge St^{\mathbb{R}})))$$

What can we do if a sketch $\mathbb{G}$ does not satisfy a general constraint $(c, !_G)$ for a simple condition of the form $c = \mathbf{0} \blacktriangleright \forall(L : (\bigwedge St^1 \to \exists(\varphi, R : \bigwedge St^2)))$ where St^1 is a set of statements in L and St^2 a set of statements in R, respectively?

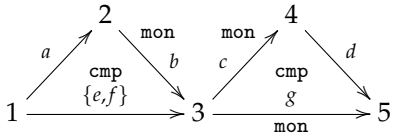
We can repair this flaw by applying the sketch arrow $\varphi : (L, St^1) \to (R, St^2 \cup \varphi(St^1))$ as a transformation rule for all sketch morphisms $\tau : (L, St^1) \to \mathbb{G}$ not satisfying the conclusion in condition c. In other words, a match of the transformation rule is given by a context morphism $\tau : L \to G$ such that $\mathbb{G} \models (\bigwedge St^1, \tau)$ and $\mathbb{G} \models (\neg\exists(\varphi, R : \bigwedge St^2), \tau)$. Note that the negative application condition $\mathbb{G} \models (\neg\exists(\varphi, R : \bigwedge St^2), \tau)$ ensures that we do not apply the rule twice for the same match $\tau : (L, St^1) \to \mathbb{G}$. Applying the rule φ via the match τ means nothing but to construct a pushout in the category Sk_s^a of sketches and strict sketch arrows (compare Remark 32).

$$\begin{array}{ccc} (L, St^1) & \xrightarrow{\varphi} & (R, St^2 \cup \varphi(St^1)) \\ {\scriptstyle\tau}\downarrow & PO & \downarrow{\scriptstyle\tau^*} \\ \mathbb{G} & \xrightarrow[\varphi^*]{} & \mathbb{H} \end{array}$$

Depending on the properties of the context morphism $\varphi : L \to R$ the pushout construction may have different effects. The context G can be extended and/or factorized and, if $St^2 \neq \emptyset$, we will add new statements to the statements originating from $\mathbb{G}$.

In terms of sketch constraints, we can describe the crucial effect of the rule application as follows: $\mathbb{H}$ satisfies the constraint $(\bigwedge St^2, \tau^*)$ in addition to the constraint $(\bigwedge St^1, \tau; \varphi^*)$ inherited from $\mathbb{G}$.

Example 52 (Repairing $\mathtt{Stm}_{CT}$-sketches)**.** *As discussed in Example 51, there is one violation of the general constraints* $(\overline{guc}, !_G)$ uniqueness of composition *by the* $\mathtt{Stm}_{CT}$*-sketch* $\mathbb{G} = (G, St^{\mathbb{G}})$ *in Example 49 and one violation of* $(ct_3, !_G)$ decomposition of monomorphisms*. Repairing these two violations by pushout constructions, as described above, will result in a* $\mathtt{Stm}_{CT}$*-sketch* $\mathbb{H}$ *like the one visualized below.*



$\mathbb{G}$ *also does not satisfy the general constraint* $(\overline{gec}, !_G)$ definedness of composition *and the general constraint* existence of identities *that has not been formalized in Example 51. We do not want to require that any* $\mathtt{Stm}_{CT}$*-sketch satisfies these two general constraints since we do not intend to use* $\mathtt{Stm}_{CT}$*-sketches just as encodings of categories but rather as (hopefully finite) representations of (possibly infinite) categories. This is the original purpose of sketches in category theory. See also the later discussion in Remark 40.*

Deduction

Generating new statements from given statements by means of rules is the essence of deduction in logic. An interesting observation is that the "repairing procedure", discussed in the last paragraph, can be also described as a procedure deducing new sketch constraints from given sketch constraints.

We consider a sketch $\mathbb{G}$ together with a set $C^{\mathbb{G}}$ of sketch constraints on G. If $C^{\mathbb{G}}$ contains a general constraint $(c, !_G)$ with $c = \mathbf{0} \blacktriangleright \forall(L : (\bigwedge St^1 \to \exists(\varphi, R : \bigwedge St^2)))$, we can deduce a local sketch constraint $((\bigwedge St^1 \to \exists(\varphi, R : \bigwedge St^2)), \tau)$ on G for any context morphism $\tau : L \to G$. This step corresponds to the **universal elimination rule** in classical first-order logic.

We do have a sound "quasi-propositional" **modus ponens rule schemata** for sketch constraints at hand: For all contexts X, all sketch conditions $X \blacktriangleright c_1$, $X \blacktriangleright (c_1 \to c_2)$ and all context morphisms $\mu : X \to Y$, the sketch constraints (c_1, μ) and $((c_1 \to c_2), \mu)$ imply the sketch constraint (c_2, μ).

If there is a constraint $(\bigwedge St^1, \tau) \in C^{\mathbb{G}}$, we can apply this modus ponens rule and deduce the sketch constraint $(\exists(\varphi, R : \bigwedge St^2), \tau)$ on G. Keep in mind that $L \blacktriangleright \exists(\varphi, R : \bigwedge St^2)$! The pushout construction generates, finally, the constraint $(\bigwedge St^2, \tau^* : R \to H)$ on H. This looks very much like an analogon to **Skolemization** in classical first-order logic. More precisely, we can consider this pushout construction as a pendant to the introduction of Skolem constants. This is quite in accordance with the characterization of operations in graph term algebras by pushouts in [3].

As another example, motivating the use of sketch constraints as "first class citizens", we discuss atomic Ξ_{CT}-statements, as introduced and discussed in the Examples 39 and 49: We included now the feature symbols `mon` and `fnl` in our sample footprint Ξ_{CT} to exemplify, in a more appropriate way, the use of feature symbols in diagrammatic specifications in general. In Example 51, we discussed, first, that we can specify known or desired properties of features by means of sketch conditions. Later, we have shown that we can even express the universal properties, defining the concepts "monomorphism" and "final object", respectively, by means of sketch conditions.

Given a $\mathtt{Stm}_{CT}$-sketch $\mathbb{G} = (G, St^{\mathbb{G}})$, the sketch condition $\overline{mon}$, defining the concept monomorphism, may help us to deduce from the `cmp`-statements, present in $St^{\mathbb{G}}$that two parallel edges in G have to be identified. We need just a rule which generates for each atomic Ξ_{CT}-statement $(\alpha(\mathtt{mon}), \mathtt{mon}(id_{\alpha(\mathtt{mon})}), \beta : \alpha(\mathtt{mon}) \to G)$ in $St^{\mathbb{G}}$ a corresponding sketch constraint $(\overline{mon}, \beta)$ on $\mathbb{G}$. This works so easy, since we designed our examples

in such a way that the context of $\overline{mon}$ is just $\alpha(\mathtt{mon})$. In general, any atomic Ξ-statement $(\alpha(\mathtt{mon}), \mathtt{mon}(id_{\alpha(\mathtt{mon})}), \beta)$ $(\alpha(P), P(id_{\alpha(P)}), \beta : \alpha(P) \to G)$ and any condition $K \blacktriangleright c$ may generate a sketch constraint $(c, \gamma; \beta)$ for any context morphism $\gamma : K \to \alpha(P)$. Since β binds all "free variables" in $\overline{mon}$, we just need to adapt the three steps (1) universal elimination, (2) modus ponens and (3) Skolemization, as discussed above for general constraints, to deduce identifications of parallel edges in G.

To keep Ξ_{CT} as small as possible, we have not included in Ξ_{CT} feature symbols for other limits and colimits like `equ`, `pb`, `po`, `prod`, for example. Employing sketch constraints we can even avoid to do this! As discussed in Remark 10, any (co)limit of shape I is axiomatized by the feature Ξ_{CT}-expressions $exists_I$ and $unique_I$.

Analogous to "anonymous functions" in programming, we can use the sketch condition $C_I \blacktriangleright \bigwedge\{tr_{C_I}(exists_I), tr_{C_I}(unique_I)\}$ as an **anonymous feature** representing the (co)limit concept that corresponds to the shape graph I. With anonymous features, we can not formulate statements, i.e., entities within a sketch $\mathbb{G}$, but constraints, like $(\bigwedge\{tr_{C_I}(exists_I), tr_{C_I}(unique_I)\}, \beta : C_I \to G)$ on the sketch $\mathbb{G}$.

There should be now sufficient evidence that it will be beneficial to work in future DPF with sketch constraints as first class citizens and our discussion suggests, especially, to employ pairs of a sketch $\mathbb{G} = (G, St^{\mathbb{G}})$ and a set $C^{\mathbb{G}}$ of sketch constraints on $\mathbb{G}$ as an appropriate formalization of software models. We call those pairs $((G, St^{\mathbb{G}}), C^{\mathbb{G}})$ **constrained sketches**.

Remark 40 (Constrained sketches in MDE)**.** *Our approach to use and develop DPF as a theoretical foundation of MDE is based on the idea that any diagrammatic specification formalism/technique is characterized by a certain choice of a category* Cxt *and a footprint* Ξ *where the corresponding diagrams/models can be described as* `Stm`*-sketches. Sketch conditions and sketch constraints have been developed to provide the necessary additional means to describe/constrain the syntactic structure of diagrams/models. In such a way, we can characterize now a diagrammatic specification formalism not only by a certain category* Cxt *and a certain footprint* Ξ *but also by an additional set of* `Stm`*-sketch conditions.*

We should, however, distinguish between two kinds of `Stm`*-sketch conditions: The first kind of conditions is used to formulate those constraints on* `Stm`*-sketches* $\mathbb{G}$ *that can be legally used as elements in* $C^{\mathbb{G}}$*. For a constrained* `Stm`*-sketch* $(\mathbb{G}, C^{\mathbb{G}})$*, the occurrence of a constraint* (c, τ) *in* $C^{\mathbb{G}}$ *will certify that* $\mathbb{G} \models (c, \tau)$*. Requirements for the* relational data model *[18,21] like "every table must have a primary key" and "a foreign key should only refer to a primary key" will be formalized by conditions of this kind.*

Conditions formalizing requirements like "inheritance is transitive" or "a subclass inherits all attributes of all its superclasses", however, should not be included in any $C^{\mathbb{G}}$ *to avoid diagrams/models becoming too polluted with redundant information. Those additional conditions are part of the formalism as a whole and represent the background knowledge and rules that can be used to deduce for any constrained sketch information from the information given in* $St^{\mathbb{G}}$ *and* $C^{\mathbb{G}}$*, respectively, and to repair violations of the constraints in* $C^{\mathbb{G}}$*.*

Conceptual Hierarchy

Introducing constrained sketches teleports us "back to start" but on a higher conceptual level: We do have a category Sk_s^a of sketches. To any sketch $\mathbb{G} = (G, St^{\mathbb{G}})$, we can assign the set $\mathtt{Cstr}(\mathbb{G})$ of all sketch constraints $(c, \tau : K \to G)$ on context G with c a first-order sketch condition in $\mathtt{SC}(K)$ according to Definition 27. Analogously to the translation of statements in Institutions of Statements, we can define for any sketch morphism $\varphi : \mathbb{G} \to \mathbb{H}$ a map $\mathtt{Cstr}(\varphi) : \mathtt{Cstr}(\mathbb{G}) \to \mathtt{Cstr}(\mathbb{H})$ by simple post-composition with the underlying context morphism $\varphi : G \to H$: $\mathtt{Cstr}(\varphi)(c, \tau) := (c, \tau; \varphi)$ for all $(c, \tau) \in \mathtt{Cstr}(\mathbb{G})$. This gives us trivially a functor $\mathtt{Cstr} : \mathsf{Sk}_s^a \to \mathsf{Set}$ at hand.

This situation is, however, just an instance of the abstract pattern we started with in this section: The category Sk_s^a can be taken as an instance of Ct and the functor $\mathtt{Cstr} : \mathsf{Sk}_s^a \to \mathsf{Set}$ as an instance of $\mathtt{St} : \mathsf{Ct} \to \mathsf{Set}$, respectively. The constrained sketches are then nothing but

the "abstract sketches" for this instance! We can now consider first-order sketch conditions and sketch constraints for this new instance and will finally obtain a further instance of the "abstract pattern". Potentially, we can even iterate this procedure ad infinitum.

Iterating this procedure is maybe not that relevant for DPF at the moment. We take it, however, as a good sign that our category independent approach allows us to move in and furnish the next higher level in the conceptual hierarchy whenever it is necessary and/or opportune.

Remark 41 (Makkai's Hierarchy of Sketches)**.** *We continue the discussion in Remark 29 and rise the question: How is our* conceptual hierarchy *related to the "hierarchy of sketches" in [15]?*

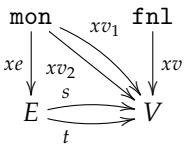
Makkai considers only atomic statements and starts with a presheaf topos, i.e., a functor category $\mathsf{Base} = \mathsf{Var} = \mathsf{Cxt} = [\mathsf{C} \to \mathsf{Set}]$*. Note that topoi are adhesive [44]! As an example, we consider the presheaf topos:*

$$\mathsf{Graph} \cong [\ id_E \circlearrowright E \underset{t}{\overset{s}{\rightrightarrows}} V \circlearrowright id_V \ \longrightarrow \mathsf{Set}].$$

Then, he describes an instance of a general construction in Category Theory: For any footprint $\Xi = (\Phi, \alpha)$*,* $\alpha \colon \Phi \to [\mathsf{C} \to \mathsf{Set}]_{Obj}$*, there is a category* $\Phi\vec{\alpha}\mathsf{C}$ *such that the category* mSk^a_s *of multi* $\mathtt{At}$*-sketches (see Remark 34) is isomorphic to the presheaf topos* $[\Phi\vec{\alpha}\mathsf{C} \to \mathsf{Set}]$*.*

$\Phi\vec{\alpha}\mathsf{C}$ *can be constructed as follows: We take the disjoint union of* Φ *(as a discrete category) and* C*. For any feature symbol* $P \in \Phi$*, any object* C *in* C*, and any* $c \in \alpha(P)(C)$*, we add an arrow* $(P, c, C) : P \to C$*. Finally, we define the composition for the new pairs of composable arrows:* $(P, c, C); f := (P, \alpha(P)(f)(c), C')$ *for all* $f : C \to C'$ *in* C*.*

As an example, we take $\Phi = \{\mathtt{mon}, \mathtt{fnl}\}$ *with arities as in Example 49. The category* $\Phi\vec{\alpha}\mathsf{Graph}$ *is visualized below. Composition in* $\Phi\vec{\alpha}\mathsf{Graph}$ *is defined by the equations* $xe; s = xv_1$*,* $xe; t = xv_2$ *and these equations encode the arity* $xv_1 \overset{xe}{\to} xv_2$ *of* $\mathtt{mon}$*! The isomorphism transforms any multi* $\mathtt{At}$*-sketch* $\mathbb{K} = (K, I^{\mathbb{K}}, st^{\mathbb{K}})$ *into a corresponding functor* $\mathcal{K} : \Phi\vec{\alpha}\mathsf{Graph} \to \mathsf{Set}$*.*



$\mathcal{K}(\ E \underset{t}{\overset{s}{\rightrightarrows}} V\ )$ *represents the graph K. The set* $\mathcal{K}(\mathtt{mon})$ *holds all the identifiers* $i \in I^{\mathbb{K}}$ *with* $st^{\mathbb{K}}(i) = (\alpha(\mathtt{mon}), \mathtt{mon}(id_{\alpha(\mathtt{mon})}), \beta)$ *while the maps* $\mathcal{K}(xe)$*,* $\mathcal{K}(xv_1)$*,* $\mathcal{K}(xv_2)$ *encode all the corresponding bindings* $\beta : \alpha(\mathtt{mon}) \to K$*. Morphisms in* $[\Phi\vec{\alpha}\mathsf{C} \to \mathsf{Set}]$*, i.e., natural transformations, encode strict* $\mathtt{At}$*-sketch arrows between multi* $\mathtt{At}$*-sketches* $\mathbb{K} = (K, I^{\mathbb{K}}, st^{\mathbb{K}})$*.*

After transforming mSk^a_s *into* $[\Phi\vec{\alpha}\mathsf{C} \to \mathsf{Set}]$*, we can define another footprint* $\Xi' = (\Phi', \alpha')$*,* $\alpha' : \Phi' \to [\Phi\vec{\alpha}\mathsf{C} \to \mathsf{Set}]_{Obj}$ *on this next level of the hierarchy and start again but this time with atomic* Ξ'*-statements.*

There are no sketch conditions in [15] but any multi $\mathtt{At}$*-sketch* $\mathbb{K} = (K, I^{\mathbb{K}}, st^{\mathbb{K}})$ *corresponds to the* $\mathtt{At}$*-sketch condition* $K \blacktriangleright \bigwedge\{st^{\mathbb{K}}(i) \mid i \in I^{\mathbb{K}}\}$ *and any strict* $\mathtt{At}$*-sketch arrow* $\varphi : \mathbb{L} \to \mathbb{R}$ *corresponds to a* $\mathtt{At}$*-sketch condition of the form* $\mathbf{0} \blacktriangleright \forall(L : (\bigwedge St^{\mathbb{L}} \to \exists(\varphi, R : \bigwedge St^{\mathbb{R}})))$*. As we have seen, sketch conditions of this special form, and thus strict* $\mathtt{At}$*-sketch arrows, allow us to axiomatize arbitrary limits or colimits, respectively.*

In such a way, all the arities $\alpha'(P')$ *in the footprint* Ξ' *correspond to very simple* $\mathtt{At}$*-sketch conditions that are just conjunctions of* $\mathtt{At}$*-statements and atomic* $\mathtt{At}'$*-statements are simply conjunctions of those conjunctions of* $\mathtt{At}$*-statements, which are introduced by the arities* $\alpha'(P')$ *and obtained the "label"* P'*.*

As an example, we consider the footprint $\Xi = (\Phi, \alpha)$*,* $\alpha \colon \Phi \to \mathsf{Graph}_{Obj}$ *with* $\Phi = \{\mathtt{cmp}, \mathtt{id}\}$ *and arities as in Example 14. For the footprint* $\Xi' = (\Phi', \alpha')$*,* $\alpha' : \Phi' \to [\Phi\vec{\alpha}\mathsf{Graph} \to \mathsf{Set}]_{Obj}$*, we assume that, for any* $P' \in \Phi'$*, the arity* $\alpha'(P')$ *corresponds to an* $\mathtt{At}$*-sketch that represents one of the commutative (co)cones described in Remark 10. In such a way, an atomic* $\mathtt{At}'$*-sketch represents a graph with a set of commutative (co)cones labelled by feature symbols from* Φ'*. Strict atomic*

$\mathtt{At}$*'-sketch arrows should allow us then to formulate propositions like: If we have binary products and equalizers, do we also have pullbacks!?*

We close this remark with a revision of the concept of graph in [22]: For the footprint $\Xi_R = (\mathsf{Rel}, \alpha)$ *in Example 50, we can consider* α *as a map* $\alpha : \mathsf{Rel} \to [1 \to \mathsf{Set}]_{Obj}$ *with* V *the only object in* 1 *and* $\alpha(P)(V) = \{x_1, x_2\}$ *for all* $P \in \mathsf{Rel}$. $\Phi\vec{\alpha}1$ *contains then for each* $P \in \mathsf{Rel}$*:*

$$\cdots \quad P \underset{(P,x_2)}{\overset{(P,x_1)}{\rightrightarrows}} V$$

an "edge sort" P *and* $[\Phi\vec{\alpha}1 \to \mathsf{Set}]$ *is the category of graphs with an* Rel*-indexed family of edges. This category is adhesive in contrast to the category of* Rel*-labelled graphs in [22]!* □

7. Conclusions

The paper presents an abstract framework allowing us to construct, in a uniform and universal way, specification formalisms in arbitrary categories enabling us to specify semantic structures while employing the full expressive power of first-order logic.

The framework is based upon a formalization of "open formulas" as statements in contexts and offers a freshly new and abstract view of logics and specification formalisms.

Relying on the new framework, we present a general and universal account of "syntactic" encodings and representations of semantic structures generalizing the idea of elementary diagrams in traditional first-order logic.

Guided by the top-down principle, we consider at this first stage of extension of our framework just simple categories. To extend a specification formalism to a proper logic, we also have to develop, however, appropriate deduction calculi. To establish those deduction calculi, we should have features, like the translation of statements along variable substitutions, for example, at hand. As exemplified in the paper, we have to assume at least the existence of pushouts to support those features. We are not logicians, but the extension of our framework by general deduction calculi will be one of the main topics in our future work.

Another main topic will be operations. At the present stage, our abstract framework does not comprise operations since it is not clear for us how to generalize the concept of operation from set-based structures to semantic structures defined in an arbitrary category. Already, the step from operations on sets to operations on graphs is not that trivial, and even the concepts, constructions and results we developed for graph operations in [3] are not fully satisfactory yet.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Acknowledgments: I want to thank the guest editor of this special volume for encouraging me to write this paper.

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

Appendix A. Translation of Feature Expressions

For a footprint Ξ and an object X in Var we denote by $\mathtt{FE}(\Xi, X)$ the set of all feature Ξ-expressions on X. In Example 26, we discussed the replacement of auxiliary feature symbols by feature expressions. To formalize those replacements, we consider footprint morphisms. A **morphism** $\eta : \Xi \to \Xi'$ between two footprints over the same category Var is given by a map η assigning to each feature symbol $F \in \Phi$ a feature Ξ'-expression $\eta(F) \in \mathtt{FE}(\Xi', \alpha(F))$. η is called **simple** if $\eta(F) = F'(id_{\alpha(F)})$, with $F' \in \Phi'$ and $\alpha'(F') = \alpha(F)$, for all $F \in \Phi$.

Any footprint morphism $\eta : \Xi \to \Xi'$ induces an Var_{Obj}-indexed family of maps $\eta_X : \mathtt{FE}(\Xi, X) \to \mathtt{FE}(\Xi', X)$. To define these maps for non-simple footprint morphisms, we have to rely, however, on a mechanism translating feature expressions along variable

translations. Fortunately, we can establish such a mechanism, if Var has pushouts, and we fix a choice of pushouts in Var.

Definition A1 (Translation maps)**.** *We define inductively and in parallel a family of* ***translation maps*** $\psi_\Xi : \mathtt{FE}(\Xi, X) \to \mathtt{FE}(\Xi, Z)$ *with* ψ *ranging over all variable translations* $\psi : X \to Z$*:*

1. *Atomic:* $\psi_\Xi(X \triangleright F(\beta)) := Z \triangleright F(\beta; \psi)$.
2. *Everything:* $\psi_\Xi(X \triangleright \top) := Z \triangleright \top$.
3. *Void:* $\psi_\Xi(X \triangleright \bot) := Z \triangleright \bot$.
4. *Conjunction:* $\psi_\Xi(X \triangleright (Ex_1 \wedge Ex_2)) := Z \triangleright (\psi_\Xi(Ex_1) \wedge \psi_\Xi(Ex_2))$.
5. *Disjunction:* $\psi_\Xi(X \triangleright (Ex_1 \vee Ex_2)) := Z \triangleright (\psi_\Xi(Ex_1) \vee \psi_\Xi(Ex_2))$.
6. *Implication:* $\psi_\Xi(X \triangleright (Ex_1 \to Ex_2)) := Z \triangleright (\psi_\Xi(Ex_1) \to \psi_\Xi(Ex_2))$
7. *Negation:* $\psi_\Xi(X \triangleright \neg Ex) := Z \triangleright \neg\, \psi_\Xi(Ex)$.
8. *Quantification:* $\psi_\Xi(X \triangleright Q(\varphi, Y : Ex)) := Z \triangleright Q(\varphi^*, Y^\varphi_\psi : \psi^*_\Xi(Ex))$

   for $Q \in \{\exists, \forall\}$ *where* $Z \overset{\varphi^*}{\to} Y^\varphi_\psi \overset{\psi^*}{\leftarrow} Y$ *is the chosen pushout of* $Z \overset{\psi}{\leftarrow} X \overset{\varphi}{\to} Y$*:*

Note that the pushout construction formalizes and generalizes the "introduction of fresh variables" in traditional FOL! If we choose the cospan $Z \overset{\psi^{-1};\varphi}{\longrightarrow} Y \overset{id_Y}{\longleftarrow} Y$, whenever ψ is an isomorphism, we ensure, especially, that $(id_X)_\Xi$ becomes the identity map on $\mathtt{FE}(\Xi, X)$. Since the composition of chosen pushouts does not result, in general, in a chosen pushout, the assignments $\psi \mapsto \psi_\Xi$ constitute only a pseudo functor from Var into Set. This may be a hint to develop future deduction calculi for Institutions of Statements rather in a fibred setting (compare [45])?

The translation $\psi_{\Xi_{CT}}(mon)$ of the universal property *mon* of monomorphisms in Example 29 along the unique graph morphism $\psi : (xv_1 \xrightarrow{xe} xv_2) \to xv \circlearrowright^{xe}$ gives us, for example, a definition of monic loops at hand.

For any footprint morphism $\eta : \Xi \to \Xi'$, we can define inductively and in parallel for all variable declarations X a **substitution map** $\eta_X : \mathtt{FE}(\Xi, X) \to \mathtt{FE}(\Xi', X)$ where the only non-trivial case is the base case :

1. *Atomic:* $\eta_X(F(\beta)) := \beta_{\Xi'}(\eta(F))$ for any $F \in \Phi$ and $\beta : \alpha F \to X$ in Var.

If η is simple, this base case degenerates, according to Definition A1, to a simple replacement of feature symbols:

1′. *Atomic′:* $\eta_X(F(\beta)) := \beta_{\Xi'}(F'(id_{\alpha(F)})) = F'(id_{\alpha(F)}; \beta) = F'(\beta)$.

Thus, we do not need to employ translation maps to define substitution maps in case of simple footprint morphisms!

References

1. Ehrig, H.; Mahr, B. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*; EATCS Monographs on Theoretical Computer Science; Springer: Berlin, Germany, 1985; Volume 6.
2. Diaconescu, R. *Institution-Independent Model Theory*; Studies in Universal Logic: Basel, Switzerland, 2008. doi:10.1007/978-3-7643-8708-2.
3. Wolter, U.; Diskin, Z.; König, H. Graph Operations and Free Graph Algebras. In *Graph Transformation, Specifications, and Nets—In Memory of Hartmut Ehrig*; Springer: Cham, Switzerland, 2018; Volume 10800, pp. 313–331. doi:10.1007/978-3-319-75396-6_17.
4. Kaphengst, H.; Reichel, H. *Algebraische Algorithmentheorie*; WIB 1; VEB Robotron, Zentrum für Forschung und Technik: Dresden, Germany, 1971.
5. Reichel, H.; Hupbach, U.R.; Kaphengst, H. *Initial Algebraic Specification of Data Types, Parameterized Data Types, and Algorithms*; Technical Report 15; VEB Robotron, Zentrum für Forschung und Technik, Dresden: Dresden, Germany, 1980.
6. Reichel, H. *Initial Computability, Algebraic Specifications, and Partial Algebras*; Oxford University Press: Oxford, UK, 1987.
7. Wolter, U. An Algebraic Approach to Deduction in Equational Partial Horn Theories. *J. Inf. Process. Cybern.* **1990**, *27*, 85–128.
8. Lawvere, F.W. Functorial Semantics of Algebraic Theories. *Proc. Natl. Acad. Sci. USA* **1963**, *50*, 869–872.
9. Claßen, I.; Große-Rhode, M.; Wolter, U. Categorical concepts for parameterized partial specifications. *Math. Struct. in Comp. Science* **1995**, *5*, 153–188. doi:10.1017/S0960129500000700.
10. Barr, M.; Wells, C. *Category Theory for Computing Science*; Series in Computer Science; Prentice Hall International: London, UK, 1990.

11. Johnson, M.; Rosebrugh, R.; Wood, R. Entity-relationship-attribute designs and sketches. *Theory Appl. Categ.* **2002**, *10*, 94–112.
12. Wells, C. *Sketches: Outline with References*; Addendum 2009; Department of Mathematics, Case Western Reserve University: Cleveland, UH, USA, 1993.
13. Diskin, Z.; Wolter, U. A Diagrammatic Logic for Object-Oriented Visual Modeling. *Electron. Notes Theor. Comput. Sci.* **2008**, *203/6*, 19–41. doi:10.1016/j.entcs.2008.10.041.
14. Ehrig, H.; Ehrig, K.; Prange, U.; Taentzer, G. *Fundamentals of Algebraic Graph Transformations*; EATCS Monographs on Theoretical Computer Science; Springer: Berlin/Heidelberg, Germany, 2006. doi:10.1007/3-540-31188-2.
15. Makkai, M. Generalized Sketches as a Framework for Completeness Theorems. *J. Pure Appl. Algebra* **1997**, *115*, 49274.
16. Cadish, B.; Diskin, Z. Heterogeneous view integration via sketches and equations. In Proceedigs of the 9th International Symposium on Methodologies for Intelligent Systems, Zakopane, Poland, 9–13 June 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 603–612. doi:10.1007/3-540-61286-6_184.
17. Diskin, Z. Towards algebraic graph-based model theory for computer science. *Bull. Symb. Log.* **1997**, *3*, 144–145.
18. Rutle, A. Diagram Predicate Framework: A Formal Approach to MDE. Ph.D. Thesis, Department of Informatics, University of Bergen, Bergen, Norway, 2010.
19. Wolter, U.; Mantz, F. *The Diagram Predicate Framework in View of Adhesive Categories*; Technical Report 358; Department of Informatics, University of Bergen: Bergen, Norway, 2013.
20. König, H.; Wolter, U. Van Kampen Colimits and Path Uniqueness. *Log. Methods Comput. Sci.* **2018**, *14*, 1–27. doi:10.23638/LMCS-14(2:5)2018.
21. Rutle, A.; Rossini, A.; Lamo, Y.; Wolter, U. A formal approach to the specification and transformation of constraints in MDE. *J. Log. Algebr. Program.* **2012**, *81/4*, 422–457. doi:10.1016/j.jlap.2012.03.006.
22. Rensink, A. Representing first-order logic using graphs. In Proceedings of the Graph Transformations, Second International Conference, ICGT 2004, Rome, Italy, 28 September–2 October 2004; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3256, pp. 319–335. doi:0.1007/978-3-540-30203-2_23.
23. Habel, A.; Pennemann, K. Correctness of high-level transformation systems relative to nested conditions. *Math. Struct. Comput. Sci.* **2009**, *19*, 245–296. doi:10.1017/S0960129508007202.
24. Bruggink, H.J.S.; Cauderlier, R.; Hülsbusch, M.; König, B. Conditional reactive systems. In Proceeding of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, Mumbai, India, 12–14 December 2011; Schloss Dagstuhl–Leibniz-Zentrum für Informatik: Wadern, Germany, 2011, Volume 13, pp. 191–203. doi:10.4230/LIPIcs.FSTTCS.2011.191.
25. Kosiol, J.; Strüber, D.; Taentzer, G.; Zschaler, S. Graph consistency as a graduated property–consistency–sustaining and-improving graph transformations. In Proceedings of the 13th International Conference, ICGT 2020, Bergen, Norway, 25–26 June 2020; Springer: Cham, Switzerland, 2020; Volume 12150, pp. 239–256. doi:10.1007/978-3-030-51372-6_14.
26. Makkai, M. First Order Logic with Dependent Sorts, with Applications to Category Theory. Available online: http://www.math.mcgill.ca/makkai/ (accessed on 31 January 2022).
27. Freyd, P.J. Properties invariant within equivalence types of categories. In *Algebra, Topology and Category Theory: A Collection of Papers in Honour of Samuel Eilenberg*; Heller, A., Tierney, M., Eds.; Academic Press: Cambridge, MA, USA, 1976; pp. 55–61.
28. Freyd, P.J.; Scedrov, A. *Categories, Allegories*; North-Holland Mathematical Library; North-Holland: Amsterdam, The Netherlands, 1990; Volume 39.
29. Wolter, U.; Klar, M.; Wessäly, R.; Cornelius, F. *Four Institutions—A Unified Presentation of Logical Systems for Specification*; Technical Report Bericht-Nr. 94-24; Fachbereich Informatik: Berlin, Germany, 1994.
30. Pawlowski, W. Context institutions. In Proceedings of the 11th COMPASS/ADT Workshop on Specification of Abstract Data Types Joint with the 8th COMPASS Workshop, Oslo, Norway, 19–23 September 1995; Springer: Cham, Switzerland, 1995; Volume 1130, pp. 436–457.
31. Goguen, J.A.; Burstall, R.M. Institutions: Abstract Model Theory for Specification and Programming. *J. ACM* **1992**, *39*, 95–146.
32. Wolter, U. Institutional frames. In Proceedings of the 10th Workshop on Specification of Abstract Data Types Joint with the 5th COMPASS Workshop, Santa Margherita Ligure, Italy, 30 May–3 June1994; Springer: Cham, Switzerland, 1995; Volume 906, pp. 469–482. doi:10.1007/BFb0014445.
33. Martini, A.; Wolter, U.; Haeusler, E.H. Fibred and Indexed Categories for Abstract Model Theory. *Log. J. IGPL* **2007**, *15*, 707–739. doi:doi:10.1093/jigpal/jzm045.
34. Wolter, U.; Martini, A.; Haeusler, E.H. Towards a uniform presentation of logical systems by indexed categories and adjoint situations. *J. Log. Comput. Oxf. Univ. Press* **2015**, *25*, 57–93. doi:10.1093/logcom/exs038.
35. McLarty, C. *Elementary Categories, Elementary Toposes*; Oxford Logic Guides (Book 21); Clarendon Press: Oxford, UK, 1991.
36. Baader, F.; Horrocks, I.; Sattler, U. Chapter 3. Description logics. In *Handbook of Knowledge Representation*; Elsevier: Amsterdam, The Netherland, 2007.
37. Goguen, J.A.; Meseguer, J. Order-sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. *Theor. Comput. Sci.* **1992**, *105*, 217–273. doi:10.1016/0304-3975(92)90302-V.
38. Ehrig, H.; Große-Rhode, M.; Wolter, U. Applications of Category Theory to the Area of Algebraic Specification in Computer Science. *Appl. Categ. Struct.* **1998**, *6*, 1–35. doi:10.1023/A:1008688122154.

39. Chang, C.C.; Keisler, H.J. *Model Theory*; Studies in Logic and the Foundations of Mathematics; Elsevier: Amsterdam, The Netherland, 1990.
40. Lloyd, J.W. *Foundations of Logic Programming*, 2nd ed.; Springer: Cham, Switzerland, 1987.
41. Wechler, W. *Universal Algebra for Computer Scientists*; EATCS Monographs on Theoretical Computer Science; Springer: Berlin, Germany, 1992; Volume 25.
42. Lack, S.; Sobociński, P. Adhesive categories. In Proceedings of the FOSSACS 2004 International Conference on Foundations of Software Science and Computation Structures, Barcelona, Spain, 29 March–2 April 2004; Volume 2987, pp. 273–288.
43. Wolter, U. Indexed vs. fibred structures—A field report. *Rom. J. Pure Appl. Math.* **2020**, *66*, 813–830.
44. Lack, S.; Sobocinski, P. Toposes are adhesive. In Proceedings of the Third International Conference, ICGT 2006, Natal, Rio Grande do Norte, Brazil, 17–23 September 2006; Springer: Cham, Switzerland, 2006; Volume 4178, pp. 184–198. doi:10.1007/11841883_14.
45. Wolter, U.; Martini, A.R.; Haeusler, E.H. Indexed and fibred structures for hoare logic. In *Electronic Notes in Theoretical Computer Science*; Elsevier: Amsterdam, The Netherland, 2020; pp. 125–145. doi:10.1016/j.entcs.2020.02.008.