Being Van Kampen in Presheaf Topoi is a Uniqueness Property*

Harald König[1] **and Uwe Wolter**[2]

1 **Department of Informatics, University of Applied Sciences FHDW Hannover, Freundallee 15, 30173 Hannover, Germany**
harald.koenig@fhdw.de

2 **Department of Informatics, University of Bergen, P.O.Box 7803, 5020 Bergen, Norway**
Uwe.Wolter@uib.no

Abstract

Fibred semantics is the foundation of the model-instance pattern of software engineering. Software models can often be formalized as objects of *presheaf topoi*, i.e, categories of objects that can be represented as algebras as well as coalgebras, e.g., the category of directed graphs. Multimodeling requires to construct *colimits* of models, decomposition is given by *pullback*. Compositionality requires an exact interplay of these operations, i.e., diagrams must enjoy the *Van Kampen* property. However, checking the validity of the Van Kampen property algorithmically based on its definition is often impossible.

In this paper we state a necessary and sufficient yet easily checkable condition for the Van Kampen property to hold in presheaf topoi. It is based on a uniqueness property of path-like structures within the defining congruence classes that make up the colimiting cocone of the models. We thus add to the statement "Being Van Kampen is a Universal Property" by Heindel and Sobociński the fact that the Van Kampen property reveals a set-based structural *uniqueness* feature.

1998 ACM Subject Classification D.3.1. Formal Definitions and Theory, F.3.2. Semantics of Programming Languages

Keywords and phrases Van Kampen Cocone, Presheaf Topos, Fibred Semantics, Mapping Path

Digital Object Identifier 10.4230/LIPIcs.CALCO.2017.16

1 Introduction

A presheaf topos is a category, that is based on an algebraic signature with unary operation symbols. Presheaves can also be considered as intersection of algebras and coalgebras [10]. Van Kampen Colimits are a generalization of Van Kampen squares [22]. In [26] we gave a necessary and sufficient condition for a pushout to be a Van Kampen square in a presheaf topos. In the present paper a corresponding criterion is given for all colimiting cocones.

1.1 Motivation

Software engineering and especially model-driven software development requires the decomposition of large models into smaller components, i.e., successful development of large

* This work was partially supported by the Department of Computing, Mathematics, and Physics at Høgskolen i Bergen

applications requires system design fragmentation. Vice versa, a comprehensive viewpoint of a related ensemble of heterogenous software-engineering components is taken up by considering the *amalgamation* (union) of these artefacts modulo their relations amongst each other. This assembly shall not only be carried out on a syntactical level (models), but in the same way on the semantical level (instances). This interplay between assembly and disassembly shows that composition and *correct* decomposition of an instance of a model into instances of the model components always accompany each other. It can be shown that correctness, i.e., *compositionality* [4, 5] is not always guaranteed [20].

Fibred semantics adheres to the model-instance pattern, a standard viewpoint in software engineering: A model M is an object of an appropriate category $\mathbb{C}$, semantics is given by the comma category $\mathbb{C}\downarrow M$. In each object $\tau \in \mathbb{C}\downarrow M$, $\tau : I \to M$, I is the instance structure and τ is its typing. Amalgamation is colimit (of the arrangement of components) and decomposition is performed by taking pullbacks along the cocone morphisms of the colimit.

To wit: Compositionality means that colimit of semantics (instances) is controlled by colimit of syntax (models) such that pullback of the instance colimit retrieves the original instances. Thus compositionality is equivalent to the *Van Kampen property* [7], an abstract characteristic which determines an exactness level for the interaction of colimits and pullbacks. It is thus often necessary to check validity of this property. However, since the definition of the property comes in terms of an equivalence of categories, see Definition 2 in the present paper, algorithmic verification based on the definition is hard even for a finite number of finite models, because the involved comma categories are infinite nevertheless.

Artefacts like UML- or ER-models are based on directed multigraphs, which in turn can be coded as a functor category $Set^{\mathbb{B}}$, where $\mathbb{B}$ has objects E (edges) and V (vertices) and non-identical arrows $s, t : E \to V$. More general metamodels, however, use more sophisticated categories $\mathbb{B}$, such as E-graphs for attributed graphs [3], bipartite graphs for Petri nets [3], or more complex structures for generalized sketches [2]. Hence, $Set^{\mathbb{B}}$ with $\mathbb{B}$ an arbitrary small category, will be the underlying category for the forthcoming investigations.

Constructing colimits in a category $\mathbb{C}$ is an operation on *diagrams*, which are usually coded as functors from a small schema category $\mathbb{I}$ to $\mathbb{C}$. In order to make our results usable for software engineering, we use the older definition for diagrams: Instead of a small category, the schema $\mathbb{I}$ is a finite multigraph and a diagram is a graph morphism from $\mathbb{I}$ to $\mathbb{C}$ [16][1]. The practical construction of colimits relies on *mapping paths*, i.e., chains of pairs of elements that are mapped to each other by the morphisms in the diagram, cf. Definition 3 in Section 3. Thus, colimit computation can easily be carried out algorithmically, if the diagram is finite and consists of finite artefacts.

Summary: While colimit construction is easy, compositionality check (validation of the Van Kampen property) is hard. The *main contribution of the present paper* is a theorem (Theorem 5 in Section 3), which states that a colimit in a presheaf topos has the Van Kampen property if and only if there are no ambiguous mapping paths between any pair of elements of the coproduct of the model artefacts. Thus the implementation of the colimit operation on the model level already provides the material for more efficient compositionality checking.

The paper is organized as follows: Section 2 introduces notation and background information, Section 3 presents the main theorem and applies it to a Software Engineering problem. Section 4 sketches the proof idea: We use a former result, in which a necessary and sufficient criterion is given for pushouts [26]. This result is translated to coequalizers and, finally, lifted to colimits of arbitrary diagrams. Section 5 concludes and discusses future research topics.

More details and elaborated proofs can be found in the underlying technical report [12].

[1] More precisely to the underlying graph of $\mathbb{C}$, see Section 2

1.2 Related Work

The Van Kampen property has its origin in algebraic topology: Topological spaces X can be investigated by a covering family of X which are related by their inclusions. Topological properties are expressed with the help of the fundamental groupoid. The Van Kampen Theorem [18] states that the colimit of the fundamental groupoids of all covering spaces is the fundamental groupoid of X, thus inferring global properties from local ones. The original idea was stated by Seifert [21] for pushouts and was further elaborated by Van Kampen [23].

Inferring global properties from local ones is the heart of sheaf theory [17]. The fibred view on sheaves is discussed in [24]. The application of Van Kampen's ideas to graphical modeling and to Software Engineering was invented in [13, 22] and then further detailed in [3] for the theory of Graph Transformations. That extensive categories and especially topoi are a reasonable playground for these theories is shown in [1, 14].

Amalgamation is a requirement for a collection of artefacts in computer science [4, 5] which has been connected to the Van Kampen property in [26]. The same property is called *exactness* in institution theory [20]. The importance of finding a feasible condition to check the Van Kampen property was caused by investigations of new methods in Graph Transformations [11, 15]. That the Van Kampen property can be characterized as a bicolimit in a comprising span bicategory [7] is a fundamental statement. Moreover, the Van Kampen property has been investigated in more special contexts [9] and can also be described with the help of weak 2-limits in CAT (https://ncatlab.org/nlab/show/van+Kampen+colimit). However, all these characterisations can hardly be applied in practice.

2 Preliminaries

This chapter recapitulates the most important notation for the following elaboration. For any category $\mathbb{C}$, $X \in \mathbb{C}$ means that X is contained in the collection of objects in $\mathbb{C}$. A *diagram* in $\mathbb{C}$ is based on a directed multigraph $\mathbb{I}$, the schema for the diagram. We write $\mathbb{I}_0$ and $\mathbb{I}_1$ for the sets of vertices and edges of $\mathbb{I}$. Formally, a diagram $\mathcal{D} : \mathbb{I} \to \mathcal{U}(\mathbb{C})$ is a graph morphism where $\mathcal{U}$ denotes the forgetful functor assigning to each category its underlying graph. For convenience reasons, however, the forgetful functor will be omitted, i.e., diagrams will be denoted $\mathcal{D} : \mathbb{I} \to \mathbb{C}$. This definition is used instead of the one, where $\mathbb{I}$ is a schema *category* rather than a graph, because it will turn out, that the results in this paper can easier be stated. The notions of (co-)cones and (co-)limits is the same modulo the adjunction $\mathcal{F} \dashv \mathcal{U}$ where $\mathcal{F}$: GRAPHS $\to$ CAT assigns to any graph its freely generated category, see [16], III, 4 for more details. Another advantage of this definition occurs in software engineering: Although the schema graph is finite, $\mathcal{F}(\mathbb{I})$ may have infinitely many arrows.

Vertices of $\mathbb{I}$ play the role of indices for diagram objects, hence, we use letters $i, j, \dots$ for vertices. Edges of $\mathbb{I}$ will be depicted $i \xrightarrow{d} j$ and we write $i = s(d)$, $j = t(d)$ (source and target of d). Images of edges under a diagram $\mathcal{D} : \mathbb{I} \to \mathbb{C}$ will be denoted $\mathcal{D}_i \xrightarrow{\mathcal{D}_d} \mathcal{D}_j$ (slightly deviating from the usual notation $\mathcal{D}(i), \mathcal{D}(d)$, etc).

Let $\mathcal{E}, \mathcal{D} : \mathbb{I} \to \mathbb{C}$ be two diagrams, then a family

$$\tau = (\tau_i : \mathcal{E}_i \to \mathcal{D}_i)_{i \in \mathbb{I}_0}$$

of $\mathbb{C}$-morphisms with $\tau_j \circ \mathcal{E}_d = \mathcal{D}_d \circ \tau_i$ for all edges $i \xrightarrow{d} j$ in $\mathbb{I}_1$ will be called a *natural transformation* between the diagrams and will be denoted in the usual way $\tau : \mathcal{E} \Rightarrow \mathcal{D}$. For any $S \in \mathbb{C}$, $\Delta S : \mathbb{I} \to \mathbb{C}$ denotes the constant diagram, which sends each edge of $\mathbb{I}$ to id_S.

S (as $\mathbb{C}$-object) and ΔS (as diagram) will be used synonymously. Diagrams together with natural transformations constitute the category $\mathbb{C}^{\mathbb{I}}$. Note that $\Delta : \mathbb{C} \to \mathbb{C}^{\mathbb{I}}$ is itself a functor, assigning to each object of $\mathbb{C}$ its constant diagram and to an arrow $f : A \to B$ the "constant" natural transformation $(f)_{i\in\mathbb{I}_0}$.

We assume all categories under consideration to have colimits. The coproduct cocone of a family $(\mathcal{D}_i)_{i\in I}$ of $\mathbb{C}$-objects will be denoted

$$(\mathcal{D}_i \xrightarrow{\subseteq_i} \textstyle\coprod_{i\in I} \mathcal{D}_i)_{i\in I}.$$

The morphisms $\subseteq_i$ are called coproduct injections. For a family of arrows $(f_i : \mathcal{D}_i \to A)_{i\in I}$ we write $\vec{f} : \coprod_{i\in I} \mathcal{D}_i \to A$ for the resulting unique mediating arrow.

We assume all categories under consideration to have pullbacks. In the sequel, we will work with *chosen pullbacks*, i.e., for each pair of $\mathbb{C}$-arrows $B \overset{h}{\to} A \overset{k}{\leftarrow} X$ a choice

$$\begin{tikzcd} Y \arrow[r, "h'"] \arrow[d, "h^*(k)"'] & X \arrow[d, "k"] \\ B \arrow[r, "h"'] & A \end{tikzcd}$$

of pullback span $(h^*(k), h')$ is determined once and for all. For all $h : B \to A$, $h^*(id_A)$ shall be chosen to be id_B. Whenever we deviate from these choices, this will be emphasized. It is well-known [6] that for fixed $h : B \to A$ chosen pullbacks along h give rise to a (pullback) functor $h^* : \mathbb{C}\downarrow A \to \mathbb{C}\downarrow B$ between comma categories. Pullbacks can be composed, i.e., if $C \overset{h_2}{\to} B \overset{h_1}{\to} A$, then $h_2^* \circ h_1^*$ yields a pullback along $h_1 \circ h_2$, and decomposed, i.e., if $h_1^*(k)$ and $(h_1 \circ h_2)^*(k)$ are computed, the resulting universal arrow from the latter into the former pullback yields a pullback of h_2 and $h_1^*(k)$. Note, that in both cases the automatically appearing pullbacks need not be chosen.

The underlying category for all further considerations is a category of *presheaves*, i.e., the category $\mathbb{G} := Set^{\mathbb{B}}$ (with $\mathbb{B}$ a small (base) category, Set the category of sets and mappings) of covariant functors from $\mathbb{B}$ to Set together with natural transformations between them[2]. We will also use the term "sort" for the objects in $\mathbb{B}$ and the term "operation (symbol)" for the morphisms in $\mathbb{B}$. It is folklore that $\mathbb{G}$ has all colimits and all pullbacks, which are computed sortwise, resp. $\mathbb{G}$ is a topos, i.e. a category with finite limits and colimits, which has exponents and where the subobject functor is representable [6]. $\mathbb{G}$ will thus also be called a *presheaf topos.* E.g., the category of multigraphs is a presheaf topos with $\mathbb{B} = (E \overset{s}{\underset{t}{\rightrightarrows}} V)$ (plus identities). The simplest presheaf topos is Set ($\mathbb{B} = 1$, the one-object-one-morphism category).

In this paper, we will make frequent use of (sortwise) coproducts, i.e., disjoint unions of sets. In order to make argumentations simpler, we will assume that for each $X \in \mathbb{B}$ the artefacts $(\mathcal{D}_i(X))_{i\in\mathbb{I}_0}$ are a priori disjoint, i.e., the coproduct is obtained by simple union.

An important property of presheaf topoi is (infinite) *extensivity*, i.e., the functor

$$\coprod : \prod_{i\in I} \mathbb{G}\downarrow D_i \to \mathbb{G}\downarrow\coprod_{i\in I} D_i \tag{1}$$

2 Normally presheaves are categories $Set^{\mathbb{B}^{op}}$, i.e., contravariant Set-valued functors. But we prefer the slightly deviating definition, because we found the contravariant version counterintuitive for our work. Clearly, it is easy to switch to the contravariant setting, if one inverts all arrows of $\mathbb{B}$.

assigning to each object $(f_i : A_i \to D_i)_{i\in I}$ in $\prod_{i\in I} \mathbb{G}{\downarrow}D_i$ the object $\coprod_{i\in I} f_i : \coprod_{i\in I} A_i \to \coprod_{i\in I} D_i$ in $\mathbb{G}{\downarrow}\coprod_{i\in I} D_i$, is an equivalence of categories for each index set I and each I-indexed family $(D_i)_{i\in I}$ of objects in $\mathbb{G}$. Its "inverse" arises from constructing pullbacks along coproduct injections. From these facts one derives the stability of coproducts under pullbacks, i.e., if

$$\begin{tikzcd} A_i \arrow[r, "a_i"] \arrow[d, "f_i"'] & A \arrow[d, "g"] \\ M_i \arrow[r, "g_i"] & M \end{tikzcd} \qquad \begin{tikzcd} \coprod_{i\in I} A_i \arrow[r, "\vec{a}"] \arrow[d, "\coprod_{i\in I} f_i"'] & A \arrow[d, "g"] \\ \coprod_{i\in I} M_i \arrow[r, "\vec{g}"] & M \end{tikzcd} \tag{2}$$

are commutative diagrams, then the squares on the left-hand side are pullbacks for all $i \in I$, if and only if the square on the right-hand side is a pullback [6]. In general topoi, all these statements still hold for finite index sets I (finite extensivity).

3 An Equivalent Condition for the Van Kampen Property

In this chapter we introduce the Van Kampen property and state the main result of this paper, a necessary and sufficient condition for the Van Kampen property to hold in $\mathbb{G} = Set^{\mathbb{B}}$.

3.1 Van Kampen Colimits

A commutative cocone out of a diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ is a natural transformation

$$\kappa : \mathcal{D} \Rightarrow \Delta S. \tag{3}$$

For fixed $i \xrightarrow{d} j$ of $\mathbb{I}_1$, pulling back a $\mathbb{G}$-arrow $K \xrightarrow{\sigma} S$ along κ_i and κ_j yields

$$\begin{tikzcd} \mathcal{E}_i \arrow[r, dashed, "\mathcal{E}_d"] \arrow[d, "\kappa_i^*(\sigma)"'] \arrow[rr, bend left, "\kappa_i'"] & \mathcal{E}_j \arrow[r, "\kappa_j'"] \arrow[d, "\kappa_j^*(\sigma)"] & K \arrow[d, "\sigma"] \\ \mathcal{D}_i \arrow[r, "\mathcal{D}_d"] \arrow[rr, bend right, "\kappa_i"'] & \mathcal{D}_j \arrow[r, "\kappa_j"] & S \end{tikzcd} \tag{4}$$

where the right and the outer rectangles are chosen pullbacks, $\mathcal{E}_d$ is the unique completion into the right pullback, and the resulting left square is a pullback by the pullback decomposition property. The left square may, however, not be a chosen one, but it results in diagram $\mathcal{E}$ as well as natural transformation $\kappa^*(\sigma) := (\kappa_i^*(\sigma))_{i\in\mathbb{I}_0} : \mathcal{E} \Rightarrow \mathcal{D}$, whose naturality squares are pullbacks. This fact gives rise to the following definition:

▶ **Definition 1** (Cartesian Transformation). A natural transformation $\tau : \mathcal{E} \Rightarrow \mathcal{D} : \mathbb{I} \to \mathbb{G}$ is called *cartesian* if all naturality squares are pullbacks.

For a fixed diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ let $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$ be the full subcategory of $\mathbb{G}^{\mathbb{I}} \downarrow \mathcal{D}$ of *cartesian* natural transformations. Thus, by (4), κ^* maps objects of $\mathbb{G} \downarrow S$ to objects in $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$. Moreover, any arrow $\gamma : \sigma \to \sigma'$ of $\mathbb{G}{\downarrow}S$ yields a family of arrows $(\kappa_i^*(\gamma))$ (universal arrows into pullbacks) of which it can easily be shown that together they yield a cartesian natural transformation $\kappa^*(\gamma) : \kappa^*(\sigma) \to \kappa^*(\sigma')$. Thus κ^* becomes a functor

$$\kappa^* : \mathbb{G}{\downarrow}S \to \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}. \tag{5}$$

▶ **Definition 2** (Van Kampen Cocone, [7]). Let $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be a diagram and $\kappa : \mathcal{D} \Rightarrow \Delta S$ be a commutative cocone. Then κ has the *Van Kampen (VK) Property* ("κ is VK") if functor κ^* is an equivalence of categories.

As usual, a *colimit* (or *colimiting cocone*) is a universal cocone $\kappa : \mathcal{D} \Rightarrow \Delta S$, i.e., for each $T \in \mathbb{G}$ and commutative cocone $\rho : \mathcal{D} \Rightarrow \Delta T$, there is a unique $\mathbb{G}$-morphism $S \xrightarrow{u} T$ such that $\Delta u \circ \kappa = \rho$, i.e., $u \circ \kappa_i = \rho_i$ for all $i \in \mathbb{I}_0$. S is called the *colimit object.*

κ^* has a left-adjoint $\kappa_* : \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} \to \mathbb{G} \downarrow S$ which assigns to a cartesian natural transformation $\tau : \mathcal{E} \Rightarrow \mathcal{D}$ the unique arrow to S out of the colimit object of the colimiting cocone of $\mathcal{E}$ [22]. I.e., κ_* is the (pseudo-)inverse of κ^*, if the VK property holds. In this case, unit and counit of the adjunction are isomorphisms. Note also that each VK cocone $\mathcal{D} \Rightarrow \Delta S$ is automatically a colimit (apply κ^* to $id_{\Delta S}$ and use the definition of κ_*) such that we can use the terms "Van Kampen cocone" and "Van Kampen colimit" synonymously.

Whereas the counit of this adjunction is always an isomorphism, if pullback functors have right-adjoints (and thus preserve colimits), which is true in every (presheaf) topos [6], the situation is more involved concerning the unit of the adjunction: The easiest example of the VK property arises for the empty diagram. In this case the property translates to the fact, that the *initial object* 0 is strict, i.e., each arrow $A \longrightarrow 0$ is an isomorphism. This is true in all topoi [6]. In the same way, since all presheaf topoi are extensive (cf. Section 2), coproducts have the Van Kampen property. But the unit fails to be an isomorphism for pushouts and coequalizers: Even in *Set* there are easy examples of pushouts which violate the VK property [22]. In *adhesive categories* (and thus in all topoi [14]) pushouts are VK, if one leg is monic, by definition. Vice versa, there are also pushouts with both legs non-monic, which enjoy this property nevertheless [26]. Astonishingly, coequalizers seldom are VK: Consider the shape graph $\mathbf{2} := \begin{tikzcd} 1 \arrow[r, shift left, "d'"] \arrow[r, shift right, "d"'] & 2 \end{tikzcd}$ and the diagram $\mathcal{D} : \mathbf{2} \to Set$ with $\mathcal{D}_1 = \{*_1\}, \mathcal{D}_2 = \{*_2\}$. Clearly,

$$\begin{tikzcd} \mathcal{D}_1 \arrow[r, shift left] \arrow[r, shift right] & \mathcal{D}_2 \arrow[r] & \{*\} \end{tikzcd} \tag{6}$$

is a coequalizer in *Set*. Then the cartesian transformation

$$\tau : \left(\begin{tikzcd} \mathcal{E}_1 := \{a,b\} \arrow[r, shift left, "id"] \arrow[r, shift right, "k"'] & \mathcal{E}_2 := \{a,b\} \end{tikzcd} \right) \Rightarrow \mathcal{D},$$

with k the non-identical bijection of $\{a,b\}$, is mapped to $id_{\{*\}}$ by κ_*, i.e., $\tau \not\cong (\kappa^* \circ \kappa_*)(\tau)$.

3.2 Equivalent Condition

As mentioned in the introduction it is important for several software engineering scenarios to find an easily checkable criterion for the Van Kampen property. The presented condition of this paper comes in terms of the mapping behavior of all morphisms $\mathcal{D}_d$ in the diagram.

▶ **Definition 3** (Mapping Path). Let $\mathbb{G} = Set^{\mathbb{B}}$ be a presheaf topos and $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be a diagram w.r.t. shape graph $\mathbb{I}$. Let $\mathbb{I}_1^{op} := \{d^{op} \mid d \in \mathbb{I}_1\}$.

- A *Path Segment* of sort $X \in \mathbb{B}$ is a triple (y, δ, y') with $\delta \in \mathbb{I}_1 \cup \mathbb{I}_1^{op}$ and[3]

If $\delta = d \in \mathbb{I}_1$	then	$y \in \mathcal{D}_{s(d)}(X), y' = \mathcal{D}_d(y) \in \mathcal{D}_{t(d)}(X)$
If $\delta = d^{op} \in \mathbb{I}_1^{op}$	then	$y' \in \mathcal{D}_{s(d)}(X), y = \mathcal{D}_d(y') \in \mathcal{D}_{t(d)}(X)$

[3] Whenever $i \xrightarrow{d} j \in \mathbb{I}_1$ and we apply a mapping in the family $((\mathcal{D}_d)_X : \mathcal{D}_i(X) \to \mathcal{D}_j(X))_{X \in \mathbb{B}}$, we write $\mathcal{D}_d$ instead of $(\mathcal{D}_d)_X$.

Two path segments (y_1, δ_1, y_1') and (y_2, δ_2, y_2') of sort X are equal if $y_1 = y_2$, $\delta_1 = \delta_2$, and $y_1' = y_2'$. Moreover, two path segments are *weakly equal*, $(y_1, \delta_1, y_1') =_w (y_2, \delta_2, y_2')$ in symbols, if $(y_1, \delta_1, y_1') = (y_2, \delta_2, y_2')$ or $(y_1, \delta_1, y_1') = (y_2', \delta_2^{op}, y_2)$.[4]

- A *Non-empty Mapping Path in $\mathcal{D}$* of sort $X \in \mathbb{B}$ is a sequence

$$P = [(y_0, \delta_0, y_1), (y_1, \delta_1, y_2), (y_2, \delta_2, y_3), \ldots, (y_{n-1}, \delta_{n-1}, y_n)]$$

  of path segments of sort X, where any third component of a segment coincides with the first component of its successor segment[5], and where $n \geq 1$. We say that the above path connects y_0 with y_n in $\mathcal{D}$.
- For each $y \in \mathcal{D}_i(X)$, where $i \in \mathbb{I}_0$ and $X \in \mathbb{B}$, we say that the *Empty Mapping Path* $[\ ]$ of sort X connects y with itself in $\mathcal{D}$.
- Two paths are equal, if they have the same length and are segmentwise equal.
- A mapping path is *proper* if there are no two distinct path segments that are weakly equal.

Examples of mapping paths for graphs are depicted in Figure 1 (the complete meaning of the contents of Figure 1 will be explained in the next section): There are two paths (one along the dashed path segments, the other one along the dotted segments) both connecting vertex ”Sort” with vertex ”Type”. Each arrow depicts a path segment with first component the arrow's source and third component its target. The middle component is annotated near the arrows, resp., their names will be explained in the next section, as well.

For any $X \in \mathbb{B}$, any $i, j \in \mathbb{I}_0$ and any $z \in \mathcal{D}_i(X)$, $z' \in \mathcal{D}_j(X)$ we write $z \equiv_X z'$ $(z \equiv_X^p z')$, if there is a mapping path (proper mapping path) of sort X connecting z with z'. It is easy to see that $\equiv\ =\ \equiv^p$ and that this relation is a congruence relation on $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ (i.e., a family of equivalence relations $(\equiv_X)_{X \in \mathbb{B}}$ compatible with operations of $\mathbb{B}$), because paths can be concatenated and reversed. Moreover, it is well-known [16] that the colimiting cocone of diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ is given by

$$\mathcal{D} \overset{\kappa}{\Rightarrow} (\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)/\equiv\ =\ (\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)/\equiv^p \tag{7}$$

where $\kappa_i = [\ ]_\equiv \circ \subseteq_i$ with $[\ ]_\equiv$ the canonical morphism. In the present paper we will show that mapping paths also play a crucial role for a simpler characterization of the Van Kampen property. The following examples hint at this connection.

▶ **Example 4.** Let $\mathbb{G} = \mathit{Set}$.

1. In (6) there are proper mapping paths $[]$ and $[(*_2, d^{op}, *_1), (*_1, d', *_2)]$ both connecting $*_2$ with itself.
2. The shape graph $1 \xleftarrow{d} 0 \xrightarrow{d'} 2$ yields pushouts. The easiest example of a non-VK pushout arises from $\mathcal{D}_0 = \{x, y\}, \mathcal{D}_1 = \{*_1\}, \mathcal{D}_2 = \{*_2\}$, cf. [22]. In this case, we obtain two different proper mapping paths $[(*_1, d^{op}, x), (x, d', *_2)]$ and $[(*_1, d^{op}, y), (y, d', *_2)]$ both connecting $*_1$ and $*_2$ in $\mathcal{D}$.
3. Let $\mathbb{I} = \begin{tikzcd} \bullet \arrow[loop left, "d"] \end{tikzcd}$ consist of one vertex and one loop. I.e., diagrams depict endomorphisms $f : A \to A$. It is astonishing that even the colimiting cocone $\mathcal{D} \Longrightarrow \{*\}$ with $\mathcal{D}_d = id_{\{*\}}$ is not VK: Take $\mathcal{E} = (\ \{a, b\} \xrightarrow{k} \{a, b\}\ )$ (with k the non-identity bijection of $\{a, b\}$),

[4] $(d^{op})^{op} := d$.

[5] By the introductory remarks on disjointness of artefacts, this means that the third component and the successor's first component are elements of the same $\mathcal{D}_i$.

$\tau : \{a, b\} \to \{*\}$, then $\mathcal{E}$'s colimit is a singleton. In this example, we have two proper mapping paths $[]$ and $[(*, d, *)]$ in $\mathcal{D}$ both connecting $*$ with itself. Note that this is just another presentation of example (6), since the colimit of $f : A \to A$ can be obtained by the coequalizer of $A \overset{id_A}{\underset{f}{\rightrightarrows}} A$).

4. $\mathcal{D} = (\{x\} \overset{g}{\underset{f}{\rightrightarrows}} \{y, z\})$ with $f(x) = y, g(x) = z$ has the VK property, which can be checked by elementary means based on Definition 2. There is exactly one proper mapping path connecting y and z, namely $(y, f^{op}, x), (x, g, z)$. Moreover, there is exactly one proper path connecting y with itself (namely the empty one, the hypothetical path $(y, f^{op}, x), (x, f, y)$ is not proper, see Definition 3). In the same way x has only one path back to itself, namely the empty one (the hypothetical path $(x, f, y), (y, f^{op}, x)$ is not proper).

As suggested by these examples, *uniqueness of proper mapping paths* between two elements of the same sort X in the sets $(\mathcal{D}_i(X))_{i \in \mathbb{I}_0}$ is a crucial feature for the Van Kampen property to hold. Indeed, we will prove

▶ **Theorem 5** (Characterization of VK Cocones as Uniqueness Property). *Let $\mathbb{G} = Set^{\mathbb{B}}$ be a presheaf topos and $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be a diagram. Let $\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$ be a colimiting cocone. The cocone is a Van Kampen cocone if and only if for all $X \in \mathbb{B}$, all $i, j \in \mathbb{I}_0$ and all $z \in \mathcal{D}_i(X)$, $z' \in \mathcal{D}_j(X)$: There are no two different proper mapping paths in $\mathcal{D}$ connecting z and z'.*

Since, in colimit computations, all mapping paths need to be computed (see (7)), and – according to Theorem 5 – the Van Kampen property can be checked by means of mapping paths, algorithmic verification of the Van Kampen property can be carried out in the background of colimit computation. In the technical report [12], we further simplify the condition of Theorem 5 and thus simplify the algorithm: We state conditions on the morphisms of $\mathcal{D}$, under which the Van Kampen property always holds (e.g., if all $\mathcal{D}_d$ are monomorphisms and enjoy a certain kind of image-disjointness) and we identify special shapes of schema graph $\mathbb{I}$, where a significantly smaller subset of indices i has to be tested for path uniqueness.

3.3 Application of Theorem 5

In order to demonstrate the benefits of this criterion, we consider a more substantial example than the ones in Example 4. We let $\mathbb{B} = (E \overset{s}{\underset{t}{\rightrightarrows}} V)$ (id_E and id_V not shown), thus our base presheaf topos is $\mathbb{G} = Set^{\mathbb{B}}$, the category of directed multigraphs. In the sequel, we depict vertices as rectangles and edges are arrows pointing from its source to its target. In Figure 1 the three highlighted graphs[6] $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ depict meta-models for type systems:

- $\mathcal{D}_1$ represents parts of the domain of *algebraic specifications*: Operations have an arbitrary number of sort-typed input parameters and exactly one return parameter.
- In $\mathcal{D}_2$ terminology of *abstract data types* is used: Functions have an arbitrary number of typed input and return parameters, resp.
- $\mathcal{D}_3$ is the object-oriented view: Interfaces own operations, which have inputs and one return parameter typed in interfaces, resp. Methods implement operations, their input parameters may be of specialized type.

[6] These are not just graphs since they contain "multiplicity constraints". They can be formalized, actually, as generalized sketches, i.e., graphs with diagrammatic predicates, in the sense of [2].

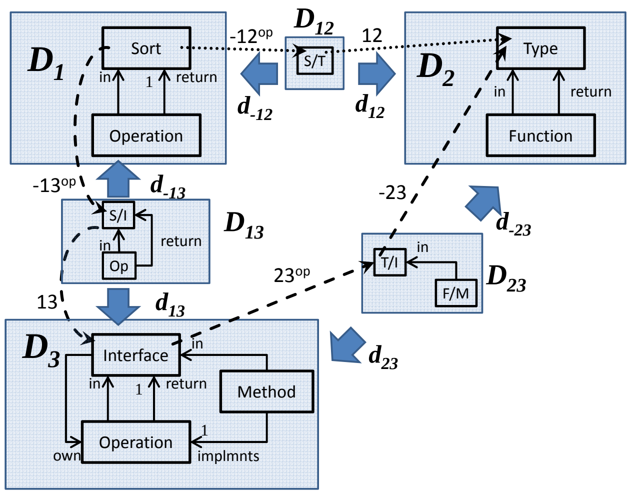

Figure 1 A diagram of metamodels and two mapping paths.

Figure 1 represents a *multimodeling* scenario [19]. Reasoning about these collective models (the multimodel) as one artefact requires matching of different terminology of each of the model graphs: Sameness of terminology in graphs $\mathcal{D}_1$ and $\mathcal{D}_2$ is formally enabled by defining a relation on $\mathcal{D}_1 \times \mathcal{D}_2$ by means of auxiliary graph $\mathcal{D}_{12}$, which consists of exactly one vertex S/T, $d_{-12}(S/T) = Sort$, $d_{12}(S/T) = Type$, such that span $\mathcal{D}_1 \overset{d_{-12}}{\leftarrow} \mathcal{D}_{12} \overset{d_{12}}{\rightarrow} \mathcal{D}_2$ specifies sameness of terms "Sort" and "Type" in graphs $\mathcal{D}_1$, $\mathcal{D}_2$ and no other commonalities. In the same way span $\mathcal{D}_1 \overset{d_{-13}}{\leftarrow} \mathcal{D}_{13} \overset{d_{13}}{\rightarrow} \mathcal{D}_3$ specifies sameness of terms "Sort" and "Interface" (in $\mathcal{D}_1$ and $\mathcal{D}_3$) as well as "Operation" (in both graphs) together with the in- and return-relationships. Moreover, relation "in" of term "Method" in $\mathcal{D}_3$ is declared to be equal to property "in" of term "Function" in $\mathcal{D}_2$ via span $\mathcal{D}_3 \overset{d_{23}}{\leftarrow} \mathcal{D}_{23} \overset{d_{-23}}{\rightarrow} \mathcal{D}_2$.

We now describe a scenario, in which colimit computation of the graphs in Figure 1 and amalgamation of instances typed over these graphs is important. It is common to reason about the multimodel by imposing constraints that spread over different models. We could, e.g., claim that "The return type of a method's implemented operation (as specified in $\mathcal{D}_3$) has to be contained in the list of return types of the corresponding function (as specified in $\mathcal{D}_2$)". In order to check this inter-model constraint, it is necessary to construct the diagram's colimit. Formally, for schema graph $\mathbb{I}$ =

$$\begin{tikzcd}
1 & & 12 \arrow[ll, "-12"'] \arrow[rr, "12"] & & 2 \\
& 13 \arrow[ul, "-13"] \arrow[dr, "13"'] & & 23 \arrow[ur, "-23"'] \arrow[dl, "23"] & \\
& & 3 & &
\end{tikzcd}$$

we obtain diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ and construct the colimiting cocone $\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$.

Assume now that we want to check consistency of given typed instances $\tau_i : \mathcal{E}_i \to \mathcal{D}_i$, $i \in \{1, 2, 3\}$ against the above formulated constraint. For this we have to declare sameness of elements within $\mathcal{E}_1, \mathcal{E}_2$, and $\mathcal{E}_3$ with the help of new relating typing morphisms $\tau_k : \mathcal{E}_k \to \mathcal{D}_k$,

$k \in \{12, 13, 23\}$ and spans $\mathcal{E}_1 \overset{e_{-12}}{\leftarrow} \mathcal{E}_{12} \overset{e_{12}}{\rightarrow} \mathcal{E}_2$, $\mathcal{E}_1 \overset{e_{-13}}{\leftarrow} \mathcal{E}_{13} \overset{e_{13}}{\rightarrow} \mathcal{E}_3$, and $\mathcal{E}_2 \overset{e_{-23}}{\leftarrow} \mathcal{E}_{23} \overset{e_{23}}{\rightarrow} \mathcal{E}_3$. Of course, all τ_k have to be compatible with model matching and sameness declaration within $\mathcal{E}_1, \mathcal{E}_2$, and $\mathcal{E}_3$, i.e., we obtain a natural transformation $\tau : \mathcal{E} \Rightarrow \mathcal{D}$ between diagrams of type $\mathbb{I} \to \mathbb{G}$. Consistency checking is then carried out by constructing the colimit object K of $\mathcal{E}$ and checking whether the resulting typing arrow $\sigma : K \to S$ fulfills the constraint, see [19].

Let us momentarily ignore constraint checking and just consider the relation between this amalgamated instance σ and the original component instances $(\tau_i)_{i \in \{1,2,3,12,13,23\}}$: It is necessary to faithfully recover all τ_i from σ, otherwise we would loose information about the origin of the elements in the domain of σ. This means that we require, for all i, $\kappa_i^*(\sigma) \cong \tau_i$, i.e., the Van Kampen property for the cocone κ has to hold. However, it turns out, that the property is violated: This can be seen by considering the following instance constellation (we write $x{:}T$ whenever $\tau\_(x) = T$): Let $\mathcal{E}_1(V) = \{s{:}Sort, s'{:}Sort\}, \mathcal{E}_1(E) = \varnothing$, $\mathcal{E}_2(V) = \{t_1{:}Type, t_2{:}Type\}, \mathcal{E}_2(E) = \varnothing$, and $\mathcal{E}_3(V) = \{\bar{i}{:}Interface, i{:}Interface\}, \mathcal{E}_3(E) = \varnothing$. One may now declare sameness of elements within $\mathcal{E}_1, \mathcal{E}_2$, and $\mathcal{E}_3$ as follows

$$s = t_1, s' = t_2 \text{ by span } (e_{-12}, e_{12}); s = i, s' = \bar{i} \text{ by } (e_{-13}, e_{13}); t_1 = \bar{i}, t_2 = i \text{ by } (e_{-23}, e_{23}).$$

This is established as described above, e.g., graph $\mathcal{E}_{12}$ consists of two vertices $1{:}S/T$ and $2{:}S/T$. Graph morphisms e_{-12} maps $1{:}S/T \mapsto s$ and $2{:}S/T \mapsto s'$ wheras e_{12} maps $1{:}S/T \mapsto t_1$ and $2{:}S/T \mapsto t_2$. We omit the obvious formal definitions of the other two spans.

Unfortunately, by transitivity, this matching also yields $s = s'$, an unwanted anomaly. But in practice this effect may happen, if two modelers work separately: One modeler might define matches (e_{-12}, e_{12}) and (e_{-13}, e_{13}) and, independently and inadvertently, the second modeler defines the match (e_{-23}, e_{23}). The inconsistent matching yields a colimit K of $\mathcal{E}$ with one vertex only, because each sort/type/interface is connected with each other along mapping paths. Clearly, $\kappa_i^*(\sigma) \ncong \tau_i$ since κ_i are monomorphisms, hence the domains of $\kappa_i^*(\sigma)$ are singleton sets, as well.

In this simple example, a small instance constellation allowed for the detection of a VK violation witness. However, it is hard to determine such witnesses in more complex examples. In these cases, Theorem 5 is a more reliable indicator for VK validity or violation, because we do not need to find violating instance constellations. Instead, the violation of the VK property can be detected by analysing mapping path structures of the metamodels only. In the present example, the indicator are the *two different proper mapping paths* of sort V shown in Figure 1 both connecting ”Sort” and ”Type” (one is depicted by dashed, the other one by dotted arrows) such that Theorem 5 immediately yields violation of the Van Kampen property. At least from this example we derive the slogan that *the Van Kampen property holds, if there is no redundant matching information* in $\mathcal{D}$. It is easy to see that the negative effect vanishes if we reduce the diagram accordingly, i.e. if we erase matching via $\mathcal{D}_{12}$ since this information is already contained in the transitive closure of matchings $\mathcal{D}_{13}$ and $\mathcal{D}_{23}$. In this way, the above mentioned modelers can indeed work independently!

4 An Outline of the Proof of Theorem 5

In this section, we sketch the main steps for the proof of our main theorem. Each step is given by a Lemma for which detailed proofs can be found in the technical report [12].

4.1 Pushouts

Often, the Van Kampen property for pushouts is formulated as follows: A pushout of a diagram $\mathcal{D}_1 \xleftarrow{h_1} \mathcal{D}_0 \xrightarrow{h_2} \mathcal{D}_2$ is said to have the Van Kampen property if for any commutative cube

$$\begin{tikzcd}
 & \mathcal{E}_0 \arrow[rr] \arrow[dl] \arrow[dd] & & \mathcal{E}_2 \arrow[dl] \arrow[dd] \\
\mathcal{E}_1 \arrow[rr] \arrow[dd] & & K \arrow[dd, "\sigma"] & \\
 & \mathcal{D}_0 \arrow[rr, "h_2"] \arrow[dl, "h_1"'] & & \mathcal{D}_2 \arrow[dl, "\kappa_2"] \\
\mathcal{D}_1 \arrow[rr, "\kappa_1"'] & & S &
\end{tikzcd}$$

with this pushout in the bottom and back faces pullbacks, the front faces are pullbacks if and only if the top face is a pushout. In [26] we already stated a characterization of the Van Kampen property for pushouts based on this definition. It comes in terms of cyclic mapping structures within $\mathcal{D}_0$:

▶ **Definition 6** (Domain Cycle, [15]). Consider a span $\mathcal{D}_1 \xleftarrow{h_1} \mathcal{D}_0 \xrightarrow{h_2} \mathcal{D}_2$ in $\mathbb{G} = Set^{\mathbb{B}}$. For $X \in \mathbb{B}$ we call a sequence $[x_0, x_1, \ldots, x_{2k+1}]$ of elements of $\mathcal{D}_0(X)$ a *domain cycle* (for the span (h_1, h_2)) (of sort X), if $k \in \mathbb{N}$ and the following conditions hold:

1. $\forall j \in \{0, 1, \ldots, 2k+1\} : x_j \neq x_{j+1}$
2. $\forall i \in \{0, \ldots, k\} : h_1(x_{2i}) = h_1(x_{2i+1})$
3. $\forall i \in \{0, \ldots, k\} : h_2(x_{2i+1}) = h_2(x_{2i+2})$

where $2k + 2 := 0$. A domain cycle is *proper* if $x_i \neq x_j$ for all $0 \leq i < j \leq 2k + 1$.

The main outcome of [26] is the following fact:

▶ **Lemma 7** (Condition for VK Pushouts). *A pushout*

$$\begin{tikzcd}
\mathcal{D}_0 \arrow[r, "h_2"] \arrow[d, "h_1"'] & \mathcal{D}_2 \arrow[d, "\kappa_2"] \\
\mathcal{D}_1 \arrow[r, "\kappa_1"'] & S
\end{tikzcd}$$

in $\mathbb{G} = Set^{\mathbb{B}}$ is a Van Kampen cocone iff there is no proper domain cycle for (h_1, h_2). ◀

This result can be proven by means of elementary set-based arguments [15], but also by investigating forgetful functors between categories of descent data [8] for general topoi [26].

It is easy to see that the above definition for pushouts is an instance of the general definition of Van Kampen colimits in Definition 2:

- If the front faces are pullbacks, then the back faces are the result of applying κ^*. Then the counit $\varepsilon : \kappa_* \circ \kappa^* \Rightarrow Id$ of adjunction is an isomorphism if and only if the cube's top face is already a pushout.
- If the top face is a pushout, then (up to isomorphism) σ is the result of applying κ_*. Then the unit $\eta : Id \Rightarrow \kappa^* \circ \kappa_*$ is an isomorphism if and only if $\kappa^*(\sigma)$ produces the original cube up to isomorphism, i.e., the original front faces are pullbacks.

Hence, the two implications ”Front face pullbacks iff top face pushout” actually reflect the two statements ”The counit is an isomorphism” and ”The unit is an isomorphism”.

Hence, Lemma 7 is a good starting point for the proof of Theorem 5: We transfer the knowledge to special mapping paths in coequalizer diagrams (Section 4.2) and from there to mapping paths in arbitrary colimits (Section 4.3).

4.2 From Pushouts to Coequalizers

The transfer from pushouts to coequalizers is accomplished in two steps. The first step connects the VK property for coequalizers and pushouts:

▶ **Lemma 8.** *Let $\mathbb{G}$ be a general topos and $B \overset{g}{\underset{f}{\rightrightarrows}} D$ be two arrows in $\mathbb{G}$. Let two arrows $\kappa_D : D \to S$ and $\kappa_B : B \to S$ be given such that the diagrams*

$$\begin{tikzcd} B \arrow[r, shift left, "g"] \arrow[r, shift right, "f"'] \arrow[rr, bend right, "\kappa_B"'] & D \arrow[r, "\kappa_D"] & S \end{tikzcd} \qquad \begin{tikzcd} B+B \arrow[r, "{[f,g]}"] \arrow[d, "{[id,id]}"'] & D \arrow[d, "\kappa_D"] \\ B \arrow[r, "\kappa_B"'] & S \end{tikzcd}$$

are commutative, resp.

1. *The left diagram is a coequalizer if and only if the right diagram is a pushout.*
2. *The left diagram is a VK cocone if and only if the right diagram is.*

Proof. (1) is well-known [16]. (2) is proven by means of Definition 2, where the transfer is possible, because topoi are (finitely) extensive, cf. Section 2, and especially because of property (2). ◀

The second step establishes a connection between domain cycles and mapping paths. It comes in terms of *disjoint* mapping paths, i.e., paths P_1 and P_2 for which non of the path segments in P_1 is weakly equal[7] to a path segment in P_2. The proof is rather technical and will be omitted, see [12], Lemma 14.

▶ **Lemma 9** (Domain Cycles vs. Mapping Paths). *Let $\mathbb{G} = Set^{\mathbb{B}}$, f and g as in Lemma 8, and $X \in \mathbb{B}$. There is a proper domain cycle of sort X for $B \xleftarrow{[id,id]} B + B \xrightarrow{[f,g]} D$, if and only if there are $z, z' \in D(X)$ and two disjoint proper mapping paths connecting z and z'.* ◀

Lemmas 7, 8, and 9 yield

▶ **Corollary 10** (Condition for VK Coequalizers). *Let $\mathbb{G} = Set^{\mathbb{B}}$, let $\mathbf{2}$ be the schema graph $1 \overset{d'}{\underset{d}{\rightrightarrows}} 2$, and $\overline{\mathcal{D}} : \mathbf{2} \to \mathbb{G}$. The coequalizer diagram*

$$\begin{tikzcd} \overline{\mathcal{D}}_1 \arrow[r, shift left, "\overline{\mathcal{D}}_{d'}"] \arrow[r, shift right, "\overline{\mathcal{D}}_d"'] \arrow[rr, bend right, "\kappa_1"'] & \overline{\mathcal{D}}_2 \arrow[r, "\kappa_2"] & S \end{tikzcd}$$

has the Van Kampen property, if and only if for all $X \in \mathbb{B}$ and all $z, z' \in \overline{\mathcal{D}}_2(X)$: There are no two disjoint proper mapping paths of sort X in $\overline{\mathcal{D}}$ connecting z and z'. ◀

Recall the already made observations in Example 4, 1. and 4., which confirm this statement.

[7] Recall the definition of weak equality in Definition 3.

4.3 From Coequalizers to Colimits

It is well-known [16], that the colimit of $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ can be computed by constructing the coequalizer of

$$\begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, shift left, "\vec{id}"] \arrow[r, shift right, "\vec{\mathcal{D}}_d"'] & \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j, \end{tikzcd} \tag{8}$$

where $\vec{\mathcal{D}}_d$ and $\vec{id}$ are mediators out of the involved coproducts:

$$\begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, "\vec{\mathcal{D}}_d"] & \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j \\ \mathcal{D}_i \arrow[u, "\subseteq_{i,d}"] \arrow[r, "\mathcal{D}_d"] & \mathcal{D}_j \arrow[u, "\subseteq_j"'] \end{tikzcd} \qquad \begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, "\vec{id}"] & \coprod_{i\in\mathbb{I}_0} \mathcal{D}_i \\ \mathcal{D}_i \arrow[u, "\subseteq_{i,d}"] \arrow[r, equal] & \mathcal{D}_i \arrow[u, "\subseteq_i"'] \end{tikzcd} \tag{9}$$

(for all edges $i \xrightarrow{d} j$ in $\mathbb{I}_1$).[8]

Let $\overline{\mathcal{D}} : \mathbf{2} \to \mathbb{G}$ be the functor mapping $\mathbf{2}$ to the objects and arrows in (8), where schema graph $\mathbf{2}$ is given as before (cf. e.g. Corollary 10). Then a technical analysis shows that we can combine mapping paths of $\mathcal{D}$ with special mapping paths of $\overline{\mathcal{D}}$ (again, we omit the proof and refer to [12]):

▶ **Lemma 11.** *Let $\mathbb{G} = Set^{\mathbb{B}}$ and $X \in \mathbb{B}$. The following statements are equivalent:*

- $\forall i, j \in \mathbb{I}_0 : \forall z \in \mathcal{D}_i(X), \forall z' \in \mathcal{D}_j(X)$*: There are no two disjoint proper mapping paths in $\mathcal{D}$ connecting z and z'.*
- $\forall z, z' \in \overline{\mathcal{D}}_2(X) = \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j(X)$*: There are no two disjoint proper mapping paths in $\overline{\mathcal{D}}$ connecting z and z'.* ◀

The main part of the proof of Theorem 5 is to carry over the VK property for the coequalizer of (8) to its underlying colimiting diagram for $\mathcal{D}$.

▶ **Lemma 12.** *For $\mathbb{G} := Set^{\mathbb{B}}$, the cocone (3) is VK if and only if the cocone*

$$\begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, shift left, "\vec{id}"] \arrow[r, shift right, "\vec{\mathcal{D}}_d"'] \arrow[rr, bend right, "\overline{\kappa}'"'] & \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j \arrow[r, "\overline{\kappa}"] & S \end{tikzcd} \tag{10}$$

resulting from constructing the coequalizer in (8) is VK.

Proof. Let $\kappa^* : \mathbb{G}\downarrow S \to \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$ be the functor introduced in (5) and $\overline{\kappa}^* : \mathbb{G}\downarrow S \to \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$ be the corresponding functor for the colimiting cocone in (10). Using (1) and (2), one can show that for each cartesian $\tau : \mathcal{E} \Rightarrow \mathcal{D}$ the squares

$$\begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{E}_{s(d)} \arrow[r, "\vec{id}"] \arrow[d, "\coprod_{d\in\mathbb{I}_1} \tau_{s(d)}"'] & \coprod_{i\in\mathbb{I}_0} \mathcal{E}_i \arrow[d, "\coprod_{i\in\mathbb{I}_0} \tau_i"] \\ \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, "\vec{id}"] & \coprod_{i\in\mathbb{I}_0} \mathcal{D}_i \end{tikzcd} \qquad \begin{tikzcd} \coprod_{d\in\mathbb{I}_1} \mathcal{E}_{s(d)} \arrow[r, "\vec{\mathcal{E}}_d"] \arrow[d, "\coprod_{d\in\mathbb{I}_1} \tau_{s(d)}"'] & \coprod_{j\in\mathbb{I}_0} \mathcal{E}_j \arrow[d, "\coprod_{j\in\mathbb{I}_0} \tau_j"] \\ \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \arrow[r, "\vec{\mathcal{D}}_d"] & \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j \end{tikzcd}$$

are pullbacks, i.e., there is the assignment $\tau \mapsto (\coprod_{d\in\mathbb{I}_1} \tau_{s(d)}, \coprod_{i\in\mathbb{I}_0} \tau_i)$. It can be shown with elementary arguments that it extends to an equivalence of categories:

$$\phi : \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} \cong \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}.$$

[8] Note that in the left coproduct of (8) an object $\mathcal{D}_i$ occurs as often as there are edges d leaving i in $\mathbb{I}$. Moreover, $\subseteq_{i,d}$ in (9) denotes the embedding of $\mathcal{D}_i$ into its appropriate copy, namely the source of $\mathcal{D}_d$.

Moreover, the colimit construction principle, see (8), yields commutativity of

$$\begin{tikzcd}
& \mathbb{G}\downarrow S \arrow[dl, "\kappa^*"'] \arrow[dr, "\overline{\kappa}^*"] & \\
\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} \arrow[rr, "\phi"'] & & \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}
\end{tikzcd}$$

up to natural isomorphism, hence, by Definition 2, the desired result. ◀

4.4 Combining the Results

We are now ready to prove Theorem 5 for disjoint proper mapping paths. This follows by combining Lemma 12, Corollary 10 and Lemma 11. Afterwards we can get rid of disjointness by showing that any two proper mapping paths connecting the same two elements also admit two disjoint proper paths (probably connecting two different elements). ◀

5 Conclusion and Future Work

In general, arbitrary diagrams in arbitrary categories are not VK. Even if we restrict to presheaf topoi, many diagrams are not VK. In the paper we presented a feasible condition (Theorem 5) to check if a diagram in a presheaf topos is VK or not.

As suggested by the example in Section 3.3, modelers may well work with a non-VK-diagram (of software models), if they have a common understanding of the used natural transformation $\tau : \mathcal{E} \Rightarrow \mathcal{D}$, i.e., if they know how to avoid "twisting anomalies" as shown in the example. Hence, the natural next step will be to look for feasible conditions that a given $\tau : \mathcal{E} \Rightarrow \mathcal{D}$ is in the image of κ^*, even if the diagram is not VK. We may allow non-uniqueness of mapping paths in diagrams of models, but then paths in the diagram of instances have to be exact copies of them, i.e., path liftings from models to instances must behave like discrete fibrations. It is worth to underline that the instances we get from a given "indexed semantics" via a corresponding variant of the Grothendieck construction [25] are always contained in the image of κ^* up to isomorphism.

The ultimate goal, however, is to find a categorical counterpart for the different paths criterion (Theorem 5), which states a necessary and sufficient condition for the Van Kampen property in more general categories. Is such a condition significantly different from the bilimit condition mentioned in the introduction and the universal property in [7]?

References

1 M. Bunge and S. Lack. Van Kampen Theorems for Topoi. *Advances in Mathematics*, 179:291 – 317, 2003.

2 Z. Diskin and U. Wolter. A Diagrammatic Logic for Object-Oriented Visual Modeling. *Electr. Notes Theor. Comput. Sci.*, 203(6):19–41, 2008. `doi:10.1016/j.entcs.2008.10.041`.

3 H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformations*. Springer, 2006.

4 Hartmut Ehrig, M. Grosse-Rhode, and U. Wolter. Applications of Category Theory to the Area of Algebraic Specification in Computer Science. *Applied Categorical Structures*, 6:1–35, 1998.

5 Jose Luiz Fiadeiro. *Categories for Software Engineering*. Springer, 2005.

6 Robert Goldblatt. *Topoi: The Categorial Analysis of Logic*. Dover Publications, 1984.

7 T. Heindel and P. Sobociński. Van Kampen Colimits as Bicolimits in Span. In A. Kurz, M. Lenisa, and A. Tarlecki, editors, *Algebra and Coalgebra in Computer Science*, volume 5728 of *Lecture Notes in Comput. Sci.*, pages 335–349. Springer Berlin / Heidelberg, 2009. `doi:10.1007/978-3-642-03741-2\_23`.

8 G Janelidze and W. Tholen. Facets of Descent, I. *Appl. Categorical Structures*, 2:245–281, 1994. `doi:10.1007/BF00878100`.

9 Wolfram Kahl. Collagories: Relation-algebraic Reasoning for Gluing Constructions. *J. Log. Algebr. Program.*, 80(6):297–338, 2011. `doi:10.1016/j.jlap.2011.04.006`.

10 Wolfram Kahl. *Categories of Coalgebras with Monadic Homomorphisms*, pages 151–167. Springer, Berlin, Heidelberg, 2014. `doi:10.1007/978-3-662-44124-4_9`.

11 Harald König, Michael Löwe, Christoph Schulz, and Uwe Wolter. Van Kampen Squares for Graph Transformation. In *Graph Transformation - 7th International Conference, ICGT 2014, Held as Part of STAF 2014, York, UK, July 22-24, 2014. Proceedings*, pages 222–236, 2014. `doi:10.1007/978-3-319-09108-2_15`.

12 Harald König and U. Wolter. Van Kampen Colimits in Presheaf Topoi. Technical report, University of Applied Sciences, FHDW Hannover, 2016. URL: `http://fhdwdev.ha.bib.de/public/papers/02016-02.pdf`.

13 S. Lack and P. Sobociński. Adhesive Categories. In *Foundations of Software Science and Computation Structures (FoSSaCS '04)*, volume 2987, pages 273–288. Springer, 2004. `doi:10.1007/978-3-540-24727-2\_20`.

14 S. Lack and P. Sobociński. Toposes are Adhesive. *Lecture Notes in Comput. Sci.*, 4178:184–198, 2006. `doi:10.1007/11841883\_14`.

15 Michael Löwe. Van Kampen Pushouts for Sets and Graphs. Technical report, University of Applied Sciences, FHDW Hannover, 2010.

16 Saunders Mac Lane. *Categories for the Working Mathematician, Second edition*. Springer, 1998.

17 Saunders MacLane and Ieke Moerdijk. *Sheaves in Geometry and Logic. A first introduction to topos theory*. Springer, 1992.

18 J.P. May. *A Concise Course in Algebraic Topology*. Chicago Lectures in Mathematics. The University of Chicago Press, 1999. `doi:10.1007/978-3-642-17336-3`.

19 Mehrdad Sabetzadeh, Shiva Nejati, Sotirios Liaskos, Steve M. Easterbrook, and Marsha Chechik. Consistency Checking of Conceptual Models via Model Merging. In *Requirements Engineering Conference*, pages 221–230, 2007.

20 Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012. `doi:10.1007/978-3-642-17336-3`.

21 Herbert Seifert. Konstruktion dreidimensionaler geschlossener Räume. *Dissertation, University of Dresden*, 1931.

22 P. Soboczińsky. Deriving Process Congruences from Reaction Rules. Technical Report DS-04-6, BRICS Dissertation Series, 2004.

23 E. R. van Kampen. On the Connection between the Fundamental Groups of some Related Spaces. *American Journal of Mathematics*, 55:261 – 267, 1933.

24 A. Vistoli. Grothendieck Topologies, Fibered Categories and Descent Theory. *Fundamental Algebraic Geometry, Math. Surveys Monogr., Amer. Math. Soc., Providence, RI, 2005*, 123:1 – 104, 2005.

25 U. Wolter and Z. Diskin. From Indexed to Fibred Semantics – The Generalized Sketch File –. Reports in Informatics 361, Dep. of Informatics, University of Bergen, 2007.

26 U. Wolter and H. König. Fibred Amalgamation, Descent Data, and Van Kampen Squares in Topoi. *Applied Categorical Structures*, 23(3):447 – 486, 2015. `doi:10.1007/s10485-013-9339-2`.