Forschungsberichte der FHDW Hannover

Van Kampen Colimits in Presheaf Topoi

Harald König, Uwe Wolter

Bericht Nr.: 02016/02

Fachhochschule für die Wirtschaft Hannover
Freundallee 15
30173 Hannover
techrep@fhdw.de

UNIVERSITY OF APPLIED SCIENCES
FHDW
FACHHOCHSCHULE FÜR DIE WIRTSCHAFT
HANNOVER

Impressum

Forschungsberichte der FHDW Hannover – Veröffentlichungen aus dem Bereich Forschung und Entwicklung der FHDW Hannover.

Herausgeber: Die Professoren der FHDW Hannover
Fachhochschule für die Wirtschaft Hannover
Freundallee 15
30173 Hannover

Kontakt: techrep@fhdw.de

ISSN 1863-7043

Van Kampen Colimits in Presheaf Topoi

Harald König
University of Applied Sciences FHDW Hannover, Germany
harald.koenig@fhdw.de
Uwe Wolter
University of Bergen, Norway
Uwe.Wolter@uib.no

Fibred semantics is the foundation of the typed-instance pattern of software engineering. Multimodeling requires to construct *colimits*, decomposition is given by pullback. Compositionality requires an exact interplay of these operations, i.e. diagrams must enjoy the *Van Kampen* property. However, algorithmic validity checking of this property based on its definition is nearly impossible.

In this paper we state a necessary and sufficient yet easily checkable condition for the Van Kampen property in *presheaf topoi*. An algorithm for colimit computation can easily be enhanced, such that it simultaneously carries out this verification. Moreover, practical guidelines describe further simplification under some additional but reasonable assumptions.

1 Introduction

Van Kampen Colimits are a straightforward generalization of Van Kampen squares [27]. In [30] 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 Van Kampen Property and Presheaf Topoi

Software engineering and especially model-driven software development requires the decomposition of large models into smaller components, i.e. successful development of large applications always requires system design fragmentation. Vice versa, a comprehensive viewpoint of a related ensemble of heterogenous software-engineering components is taken up by considering the *union* of these artefacts modulo their relations among themselves. 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. This latter claim is called *compositionality* in the literature [6, 7, 25]. It can be shown that it is not always possible to guarantee compositionality [25].

Fibred semantics adheres to the typed-instance pattern, a standard viewpoint in software engineering. I.e. semantics of a model is given by a set (or a category) of mappings (or morphisms) *into* the model – the typing assignment. Formally (categorically) union is colimit (of the arrangement of components) and decomposition in the fibred setting is 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* [12], 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, which – using its original definition – is nearly impossible to be performed algorithmically.

Submitted to:
Forschungsberichte der FHDW Hannover

Artefacts like UML-models [9] and ER-models in databases are based on directed multi-graphs which in turn can be coded as a functor category $Set^{\mathbb{B}}$ where $\mathbb{B}$ has two objects E (edges) and V (vertices) and two arrows $s,t : E \to V$. More general models, however, use more sophisticated metamodels $\mathbb{B}$, such as E-graphs for attributed graphs [5], bipartite graphs for petri-nets [5], or more complex meta-models $\mathbb{B}$ for generalized sketches [4].

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 multi-graph and a diagram is a graph morphism from $\mathbb{I}$ to $\mathbb{C}$ [20, 1][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. Def.5 in Sect.3. It can easily be carried out algorithmically for finite artifacts.

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 8 in Sect.3), which states that a colimit has the Van Kampen property if and only if there are no ambiguous mapping paths between any pair of elements of the disjoint union (coproduct) of all artefacts under consideration. Thus the implementation of the colimit operation already provides the data material for efficient compositionality checking.

In order to prove this theorem we make use of a former result, in which a necessary and sufficient criterion is given for pushouts [30]. This older result is stated in Sect. 4.1, transfered to coequalizers in 4.2 and, finally, lifted to arbitrary colimits in Sect. 4.3. In order to make Theorem 8 even better available for algorithms, the ambiguity is extended from disjoint paths to arbitrary paths using a certain path reduction technique (see Sect. 5). Thus an algorithm can avoid the checking of path disjointness, hence again be accelarated by a quadratic magnitude w.r.t. to the number of paths.

This technical report also makes a note of some more practical considerations in Sect.6: Colimit construction can be simplified for special diagrams by computing mapping path relations and their corresponding quotient only partially on the entire diagram. Furthermore, in this case, the check for the Van Kampen property can also significantly be eased. For special but still sufficiently general circumstances, we obtain an efficient algorithm for colimit computation (useful for all multimodel scenarios, cf.[24]) with simultaneous verification of the Van Kampen property, which comes almost for free (without further effort) during algorithm execution.

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. A description of the main topological properties is carried out with the help of the fundamental groupoid. The Van Kampen Theorem [22] 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 bei H. Seifert [26] for pushouts and further elaborated by Van Kampen [14].

Inferring global properties from local ones is the heart of sheaf theory [19]. The fibred view on sheaves are stacks [28]. The application of Van Kampen's ideas to graphical modeling and to Software Engineering was invented in [27, 16] and then further detailed in [5] for the use in Graph Transformations. That a reasonable playground for these theories are extensive categories and especially topoi is shown in [2, 3, 17].

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

Amalgamation is a requirement for a collection of artefacts in computer science [7, 6] which has been connected to the Van Kampen property in [30]. The same property is called *exactness* in institution theory [25]. The importance of finding a feasible condition to check the Van Kampen property was caused by investigations of new methods in graph transformations [18, 15] and diagrammatic specifications [29]. The condition given in [30] for pushouts in presheaf topoi can be used in algorithms. That the Van Kampen property can be characterized as a colimit in a comprising category of spans [12] is a more fundamental statement which can hardly be applied in practice. The Van Kampen property has also been investigated in slightly more special contexts [13] and can be desribed as bilimit in *CAT* (https://ncatlab.org/nlab/show/van+Kampen+colimit).

2 Preliminaries

This chapter recapitulates the most important notation for the following elaboration. Moreover, we list some known facts that will be used throughout the paper.

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. In this paper $\mathbb{I}$ is always finite. 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[2]. 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 $\mathcal{F} : \mathit{Graphs} \to \mathit{Cat}$ assigns to any graph its freely generated category, see [20], III, 4 for more details. Another advantage of this definition occurs in applications, e.g. in Software Engineering: Although the schema graph is finite, the category freely generated from it may have infinitely many arrows.

Vertices of $\mathbb{I}$ play the role of indices for diagram objects, hence, we use letters $i, j, \ldots$ 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 $\mathcal{D}_i \xrightarrow{\mathcal{D}_d} \mathcal{D}_j$, i.e. 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 vertex of $\mathbb{I}$ to S and 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" collection $(f)_{i \in \mathbb{I}_0}$.

We assume all categories under consideration to have colimits. The coproduct cocone of a family $(\mathcal{D}_i)_{i \in \mathbb{I}_0}$ of $\mathbb{C}$-objects will be denoted

$$(\mathcal{D}_i \xrightarrow{\subseteq_i} \coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)_{i \in \mathbb{I}_0}.$$

[2] I.e. it forgets identies and composition.

The morphisms $\subseteq_i$ are called coproduct injections.

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 $B \xrightarrow{h} A \xleftarrow{k} X$ with common codomain, a choice

$$\begin{array}{ccc} Y & \xrightarrow{h'} & X \\ {\scriptstyle h^*(k)}\downarrow & & \downarrow{\scriptstyle k} \\ B & \xrightarrow[h]{} & A \end{array}$$

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 this choice, this will be emphasized. It is well-known [11] that for fixed $h : B \to A$ chosen pullbacks along h give rise to the pullback functor $h^* : \mathbb{C}{\downarrow}A \to \mathbb{C}{\downarrow}B$ between comma categories. Pullbacks can be composed, i.e. if $C \xrightarrow{h_2} B \xrightarrow{h_1} A$, 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. Another fact, which we will use throughout the paper, is that pullbacks exist in $\mathbb{C}^{\mathbb{I}}$ and can be computed indexwise, i.e. by constructing for all $i \in \mathbb{I}_0$ the pullbacks of the i-th component of the involved natural transformations [11].

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 functors from $\mathbb{B}$ to *Set* together with natural transformations between them. 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 can be shown that $\mathbb{G}$ has all finite colimits and all pullbacks, which are computed sortwise, resp. $\mathbb{G}$ is a topos and will thus also be called a *presheaf topos* [11]. E.g. the category of multigraphs is a presheaf topos with $\mathbb{B} = E \overset{s}{\underset{t}{\rightrightarrows}} V$ (plus identities on E and V). The simplest presheaf topos is $Set = Set^{\mathbb{1}}$ with the one-object category $\mathbb{1}$.

In this paper, we will make frequent use of (sortwise) coproducts, i.e. disjoint unions of sets. In order to make argumentations simpler in proofs and background discussions, 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 sets, i.e. the coproduct can be obtained by simple union. The main results (e.g. Theorem 8), however, do not make this assumption and are formulated without mentioning coproducts. This enables more direct application of our results in practice, e.g. in software engineering.

An important property of topoi (and hence presheaves) is *extensivity*, i.e. the coproduct functor

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

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 finite 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.

Furthermore, we need the following facts for topoi (I is a finite index set).

Fact 1 *[10] Let*

$$\begin{array}{ccc} A_i & \xrightarrow{a_i} & A \\ {\scriptstyle f_i}\downarrow & & \downarrow{\scriptstyle g} \\ M_i & \xrightarrow{g_i} & M \end{array} \qquad\qquad \begin{array}{ccc} \coprod_{i \in I} A_i & \xrightarrow{[a_1,\ldots,a_n]} & A \\ {\scriptstyle \coprod_{i \in I} f_i}\downarrow & & \downarrow{\scriptstyle g} \\ \coprod_{i \in I} M_i & \xrightarrow{[g_1,\ldots,g_n]} & M \end{array}$$

be commutative diagrams. 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. □

The notation $[\_]$ and $\coprod_{i\in I} \_$ for morphisms refers to universal mappings out of / between coproducts.

Fact 2 *If the squares on the left-hand side of*

$$\begin{array}{ccc} A_i & \xrightarrow{a_i} & B_i \\ {\scriptstyle f_i}\downarrow & & \downarrow{\scriptstyle g_i} \\ C_i & \xrightarrow{b_i} & D_i \end{array} \qquad\qquad \begin{array}{ccc} \coprod_{i\in I} A_i & \xrightarrow{\coprod_{i\in I} a_i} & \coprod_{i\in I} B_i \\ {\scriptstyle \coprod_{i\in I} f_i}\downarrow & & \downarrow{\scriptstyle \coprod_{i\in I} g_i} \\ \coprod_{i\in I} C_i & \xrightarrow{\coprod_{i\in I} b_i} & \coprod_{i\in I} D_i \end{array}$$

are pullbacks for all $i \in I$, then so is the right-hand side.

The latter fact follows from extensivity: As an equivalence the functor $\coprod$ preserves products. Products in $\prod_{i\in I} \mathbb{G}{\downarrow}D_i$ are given by indexwise products. Products in slice categories, however, are given by pullbacks, i.e., the diagonal $g_i \circ a_i = b_i \circ f_i$ is the product of b_i and g_i in $\mathbb{G}{\downarrow}D_i$ with projections f_i and a_i, respectively. The right diagram represents the image of the product $(b_i \times g_i)_{i\in I}$ in $\prod_{i\in I} \mathbb{G}{\downarrow}D_i$ w.r.t. $\coprod$. It becomes, in such a way, a product in $\mathbb{G}{\downarrow}(\coprod_{i\in I} D_i)$ and thus a pullback in $\mathbb{G}$. □

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 yet easily testable condition for the Van Kampen property to hold in presheaf topoi.

3.1 Van Kampen Colimits

A commutative cocone out of a diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ is given by a (cocone) object $S \in \mathbb{G}$ and a natural transformation

$$\kappa : \mathcal{D} \Rightarrow \Delta S. \tag{1}$$

Pulling back a $\mathbb{G}$-arrow $K \xrightarrow{\sigma} S$ along all morphisms of κ yields

$$\begin{array}{ccccc} & & \overset{\kappa'_i}{\frown} & & \\ \mathcal{E}_i & \overset{\mathcal{E}_d}{\dashrightarrow} & \mathcal{E}_j & \xrightarrow{\kappa'_j} & K \\ {\scriptstyle \kappa_i^*(\sigma)}\downarrow & & \downarrow{\scriptstyle \kappa_j^*(\sigma)} & & \downarrow{\scriptstyle \sigma} \\ \mathcal{D}_i & \xrightarrow{\mathcal{D}_d} & \mathcal{D}_j & \xrightarrow{\kappa_j} & S \\ & & \underset{\kappa_i}{\smile} & & \end{array} \tag{2}$$

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 the natural transformation $\kappa^*(\sigma) := (\kappa_i^*(\sigma))_{i\in\mathbb{I}_0} : \mathcal{E} \Rightarrow \mathcal{D}$. Since pullbacks of $\mathbb{G}^{\mathbb{I}}$ are computed sortwise, this can be depicted as a $\mathbb{G}^{\mathbb{I}}$-pullback square

$$\begin{array}{ccc} \mathcal{E} & \overset{\kappa'}{\Longrightarrow} & \Delta K \\ {\scriptstyle \kappa^*(\sigma)}\Downarrow & & \Downarrow{\scriptstyle \Delta\sigma} \\ \mathcal{D} & \underset{\kappa}{\Longrightarrow} & \Delta S \end{array}$$

The fact that the resulting naturality squares of $\kappa^*(\sigma)$ in (2) are pullbacks gives rise to the following definition:

Definition 3 (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 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 natural transformation $\kappa^*(\gamma) : \kappa^*(\sigma) \to \kappa^*(\sigma')$. It is cartesian due to pullback decomposition and κ^* becomes a functor

$$\kappa^* : \mathbb{G}\downarrow S \to \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} \tag{3}$$

by the mentioned universality properties.

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 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 $i \in \mathbb{I}_0$. The following definition has been given in [12].

Definition 4 (Van Kampen Cocone) *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 *if functor κ^* is an equivalence of categories.*

It is important to note that the functor 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 $\mathcal{E}$ [27]. I.e. κ_* is the (pseudo-)inverse of κ^*, if the VK property holds. Thus the VK property means that unit and counit of 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_*). Because of this, we will use the terms "Van Kampen cocone" and "Van Kampen colimit" synonymously.

This also shows that the counit is always an isomorphism, if pullback functors have right-adjoints (and thus preserve colimits), which is true in every (presheaf) topos.

The situation is more involved concerning the unit of 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 [11]. In the same way, since all topoi are extensive (cf. Sect.2), coproducts have the Van Kampen property in $\mathbb{G}$. 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 [27]. In *adhesive categories* (and thus in all topoi [17]) pushouts are VK, if one leg is monic (which, in fact, makes up the definition of adhesiveness [27]). Vice versa, there are also pushouts with both legs non-monic, which enjoy this property nevertheless [30].

Astonishingly, coequalizers seldom are VK: Consider the shape graph $\mathbb{2} := 1 \underset{d}{\overset{d'}{\rightrightarrows}} 2$ and the diagram $\mathcal{D} : \mathbb{2} \to \mathit{Set}$ with $\mathcal{D}_1 = \mathcal{D}_2 = 1$ (a singleton set), $\mathcal{D}_d = \mathcal{D}_{d'} = id_1$. Clearly,

$$1 \rightrightarrows 1 \longrightarrow 1 \tag{4}$$

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

$$\tau : (\ 2 \underset{k}{\overset{id_2}{\rightrightarrows}} 2\ ) \Rightarrow \mathcal{D}$$

with k the non-identical bijection of 2 (any set with exactly two elements) is mapped to id_1 by κ_*, i.e. $\tau \not\cong (\kappa^* \circ \kappa_*)(\tau)$.

3.2 An Elementary and Equivalent Characterization

As mentioned in the introduction it is very important for several software engineering use-cases to find an easily checkable criterion for the Van Kampen property. In this paper we give such a feasible (necessary and sufficient) condition. It comes in terms of the mapping behavior of all morphisms $\mathcal{D}_d$ in the diagram, which is the basis for colimit computation.

Definition 5 (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 $\mathbb{I}$. We set $\mathbb{I}_1^{op} := \{d^{op} \mid d \in \mathbb{I}_1\}$.*

- *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]

$$\begin{array}{lll} \textit{If } \delta = d \in \mathbb{I}_1 & \textit{then} & y \in \mathcal{D}_{s(d)}(X), y' = \mathcal{D}_d(y) \in \mathcal{D}_{t(d)}(X) \\ \textit{and} & & \\ \textit{If } \delta = d^{op} \in \mathbb{I}_1^{op} & \textit{then} & y' \in \mathcal{D}_{s(d)}(X), y = \mathcal{D}_d(y') \in \mathcal{D}_{t(d)}(X) \end{array}$$

  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 weak 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 the third component of a segment always coincides with the first component of its successor segment, and where $n \geq 1$. We say that the above path connects y_0 with y_n. If $y_0 = y_n$[5]*, the path is called* cyclic*.*

- *For each $y \in \mathcal{D}_i(X)$, where $i \in \mathbb{I}_0$ and $X \in \mathbb{B}$, there is an* Empty Mapping Path [ ] *of sort X connecting y with itself.*

- *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*

$$z \equiv_X z'$$

  if there is a mapping path of sort X connecting z with z'.

- *A mapping path is* proper *if there are no two distinct path segments that are weak equal (thus all empty paths and paths of length 1 are proper).*

- *Two paths for $X \in \mathbb{B}$ are equal if they are equal as sequences, i.e, if they have the same length and are segmentwise equal.*

According to our disjointness assumptions the coproduct $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ is obtained by sortwise unions:

$$\forall X \in \mathbb{B} : (\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)(X) := \bigcup_{i \in \mathbb{I}_0} \mathcal{D}_i(X). \tag{5}$$

In such a way, the existence of mapping paths of sort X yields a binary relation $\equiv_X$ on $(\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)(X)$.

---

[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 $(\mathcal{D}_d)_X$.

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

[5] By the introductory remarks on disjointness of artefacts, this means that y_0 and y_n are elements of the same $\mathcal{D}_i$.

A path of sort X connecting y_0 and y_n can be composed with a path of sort X connecting z_0 and z_n if $y_n = z_0$, and the composition is just given by the concatenation of both sequences. Each path connecting y_0 with y_n can be reversed by replacing each segment (y, δ, y') by (y', δ^{op}, y) and then reversing the whole sequence. Together with the existence of empty paths, this ensures that we obtain a family $\equiv = (\equiv_X)_{X \in \mathbb{B}}$ of equivalence relations on $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$.

It is a straightforward calculation to see that it is also a congruence. Let be given a $\mathbb{B}$-arrow $op : X \to Y$. Due to our disjointness assumptions and (5) we have for all $k \in \mathbb{I}_0$ and all $x \in \mathcal{D}_k(X)$

$$(\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)(op)(x) := \mathcal{D}_k(op)(x) \in \mathcal{D}_k(Y). \tag{6}$$

If $z \equiv_X z'$, then $\mathcal{D}_i(op)(z) \equiv_Y \mathcal{D}_j(op)(z')$, the connecting path of sort Y arising from concatenating translated segments $(\mathcal{D}_{s(d)}(op)(y), d, \mathcal{D}_{t(d)}(op)(y'))$ / $(\mathcal{D}_{t(d)}(op)(y), d^{op}, \mathcal{D}_{s(d)}(op)(y'))$ for all segments (y, d, y') / (y, d^{op}, y') in the path of sort X connecting z with z'. We remark that the translated path may not be proper although the original path was.

It is well-known [20] that colimits are computed sortwise in $\mathbb{G}$ by considering the relations

$$\sim_X = \{(y, \mathcal{D}_d(y)) \mid d \in \mathbb{I}_1, y \in \mathcal{D}_{s(d)}(X)\}$$

for all $X \in \mathbb{B}$. The colimit is $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ modulo the smallest congruence relation which comprises the family $(\sim_X)_{X \in \mathbb{B}}$. On the one hand $y \sim_X y'$ implies $y \equiv_X y'$. On the other hand $z \equiv_X z'$ results from a mapping path along a sequence of elements $z = z_0, z_1, \ldots, z_n = z'$ for which either $z_i \sim_X z_{i+1}$ or vice versa. This yields the first statement in

Fact 6 (Colimit Computation) *Let $\mathbb{G}$ be a presheaf topos.*

1. *The colimiting cocone of diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ is given by*

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

   where $\kappa_i = [\,]_\equiv \circ \subseteq_i$ with $[\,]_\equiv$ the canonical morphism.

2. *For $X \in \mathbb{B}$, $i, j \in \mathbb{I}_0$ and $z \in \mathcal{D}_i(X)$, $z' \in \mathcal{D}_j(X)$ let $z \equiv^p_X z'$ if there is a* proper *mapping path from z to z', then*

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

   is also a colimiting cocone.

□

The second statement follows from a path reduction procedure (see Lemma 29 in the Appendix).

Hence mapping paths are the basis for colimit computation. 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 give a hint on the connection between these concepts.

Example 7 *Let $\mathbb{G} = Set$.*

1. *In (4) there are proper mapping paths $[\,]$ and $[(*_D, d^{op}, *_B), (*_B, d', *_D)]$ in $\mathcal{D}$ connecting $*_D$ with itself.*

2. *Let* $\mathbb{I} = \; d \circlearrowright \bullet$ *consist of one object and one loop. I.e. diagrams depict endomorphisms* $f \circlearrowright A$ *. It is astonishing that even the colimiting diagram* $\mathcal{D} = 1$ *with* $\mathcal{D} = \; id \circlearrowright 1$ *is not VK: Take* $\mathcal{E} = \; k \circlearrowright 2$ *(with k the non-identity bijection of* 2*),* $\tau : 2 \to 1$*, then* $\mathcal{E}$*'s colimit is a singleton. In this example, we have two proper mapping paths* $[]$ *and* $[(*, d, *)]$ *in* $\mathcal{D}$ *both connecting the element* $*$ *of* 1 *with itself. Note that this is just another presentation of example (4), since the colimit of* $f \circlearrowright A$ *can be obtained by the coequalizer of* $A \overset{id_a}{\underset{f}{\rightrightarrows}} A$ *) (see (14)).*

3. $\mathcal{D} = (\; \{x\} \overset{g}{\underset{f}{\rightrightarrows}} \{0,1\} \;)$ *with* $f(x) = 0, g(x) = 1$ *has the VK property (can be checked by elementary means*[6]*). There is exactly one proper mapping path connecting* 0 *and* 1*, namely* $(0, f^{op}, x), (x, g, 1)$*. Moreover, there is exactly one proper path connecting* 0 *with itself (namely the empty one, the hypothetical path* $(0, f^{op}, x), (x, f, 0)$ *is not proper, see Def.5). In the same way x has only one path back to itself, namely the empty one (the hypothetical path* $(x, f, 0), (0, f^{op}, x)$ *is again 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 8 (Characterization of VK cocones) *Let* $\mathbb{G} = Set^{\mathbb{B}}$ *be a presheaf topos and* $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ *be a diagram with* $\mathbb{I}$ *a finite directed multigraph. 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'*.*

3.3 Application of Theorem 8

In order to demonstrate the benefits of this criterion, we consider a more substantial example. 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 multi-graphs. In the sequel, we depict vertices as rectangles and edges are arrows pointing from its source to its target. In Fig.1 the three highlighted graphs $\mathcal{D}_1$, $\mathcal{D}_2$, 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.[7]

---

[6] The coequalizer of f and g is $\{0,1\} \xrightarrow{!} \{x\}$. Any cartesian nat. trafo $\tau : \mathcal{E} \to \mathcal{D}$ with $\mathcal{E} = \; X \overset{g'}{\underset{f'}{\rightrightarrows}} Y$ yields $Y \cong X + X$ with f', g' the two coproduct injections and hence $Y \xrightarrow{[id_X, id_X]} X$ to be the colimit of $\mathcal{E}$. This yields a pullback over !.

[7] This is a consequence of the fact that any graph morphism $\tau_2 : E \to \mathcal{D}_2$ admits arbitrary in/return relations between functions (i.e. vertices $v \in E$ with $\tau_2(v) = Function$) and types (vertices $v \in E$ with $\tau_2(v) = Type$).

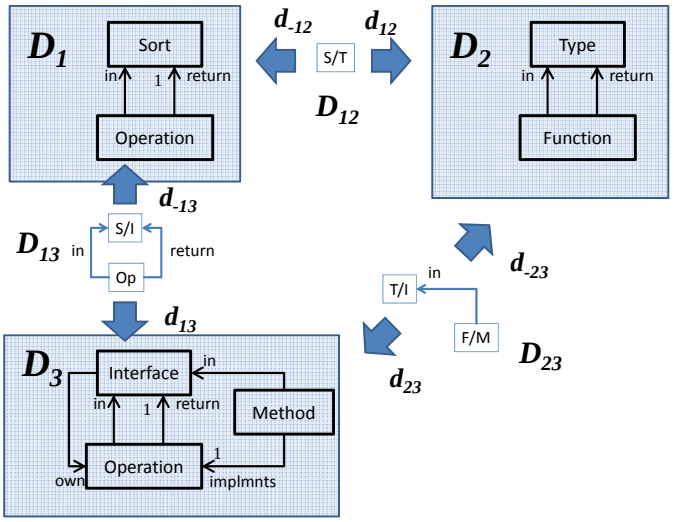


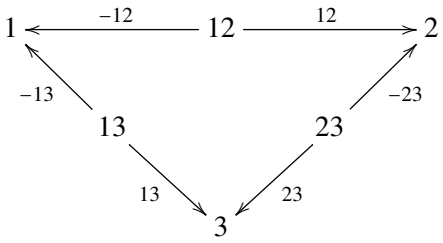
Figure 1: A colimiting cocone of a diagram of data models

- $\mathcal{D}_3$ 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.

Figure 1 represents a *multimodeling* scenario [8]. Reasoning about these collective models (the multimodel) as one artefact requires the matching of different terminology of each of the graphs: E.g. sameness of terminology in graphs $\mathcal{D}_1$ and $\mathcal{D}_2$ is enabled by defining a relation on $\mathcal{D}_1 \times \mathcal{D}_2$: Graph $\mathcal{D}_{12}$ 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 $\mathcal{D}_1$, $\mathcal{D}_2$. 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 $\mathcal{D}_2$ is declared to be equal to property "in" of term "Function" in $\mathcal{D}_3$.

Reasoning about the multimodel means to impose constraints that spread over different multimodels. E.g. we could 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 function (as specified in $\mathcal{D}_2$)". In order to check this inter-model constraint, it is necessary to construct the colimit of the 6 graphs.

Formally, for schema graph $\mathbb{I}$ =



we obtain diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ and can construct the colimiting cocone $\mathcal{D} \overset{\kappa}{\Rightarrow} S$. S together with the scope of the above mentioned constraint is shown in Fig.2.

Moreover, to check consistency of instances $\tau_i : \mathcal{E}_i \to \mathcal{D}_i$, $i \in \{1,2,3\}$ against the above formulated constraint, one has to declare sameness of instance elements with the help of morphisms $\tau_k : \mathcal{E}_k \to \mathcal{D}_k$, k =

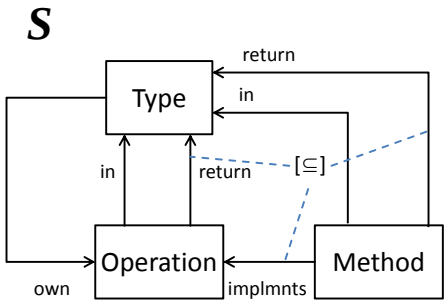


Figure 2: The colimit of the diagram from Fig.1 with a new constraint

$\{12,13,23\}$ and spans (e_{-12},e_{12}), (e_{-13},e_{13}), and (e_{-23},e_{23}) between $(\mathcal{E}_1,\mathcal{E}_2)$, $(\mathcal{E}_1,\mathcal{E}_3)$, and $(\mathcal{E}_2,\mathcal{E}_3)$, resp. Of course, typings τ_i have to be compatible with matching, 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 K of $\mathcal{E}$ and checking whether the resulting typing arrow $\sigma : K \to S$ fulfills the constraint, see [23].

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 trace back 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 $\kappa_i^*(\sigma) \cong \tau_i$, 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 in this simple example by guessing a corresponding 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 $\mathcal{E}_3(V) = \{\bar{i} : Interface, i : Interface\}, \mathcal{E}_3(E) = \varnothing$. They can be matched as follows

$$s = t_1, s' = t_2 \text{ by } \mathcal{E}_{12}; s = i, s' = \bar{i} \text{ by } \mathcal{E}_{13}; t_1 = \bar{i}, t_2 = i \text{ by } \mathcal{E}_{23}$$

But this can also happen in practice: Suppose two modelers both have a common understanding that the two sorts as well as the two types and the two interfaces must be kept seperate, resp. One modeler might define the matches $\mathcal{E}_{12}$ and $\mathcal{E}_{23}$ and, independently and inadvertently, the second modeler defines the match $\mathcal{E}_{13}$.

However, the colimit K of $\mathcal{E}$ is a graph with one vertex only, cf. the construction in Fact 6. Clearly, $\kappa_i^*(\sigma) \not\cong \tau_i$ since κ_i are monomorphisms, hence the domains of $\kappa_i^*(\sigma)$ are singletons, as well.

In more complex examples guessing of these instance constellations is much more difficult. In these cases, Theorem 8 is a more reliable indicator for VK validity or violation. In the present example, the indicator are the two different proper mapping paths

$$(Sort, d_{-13}^{op}, S/I), (S/I, d_{13}, Interface), (Interface, d_{23}^{op}, T/I), (T/I, d_{-23}, Type)$$

and

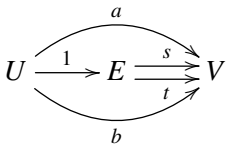
$$(Sort, d_{-12}^{op}, S/T), (S/T, d_{12}, Type)$$

of sort V from $Sort$ to $Type$ in $\mathcal{D}$.

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!

We finally remark that there are also use cases with more complex schema categories: It is useful to pass from $\mathbb{B}$ to a more complex $\mathbb{B}'$, if we consider constraints to be entities in their own right, which shall also be matched across different graphs. Consider for this Fig.2: The cardinality constraint(s) from $\mathcal{D}_1$ and $\mathcal{D}_2$ are not inherited to S, because the colimit was only constructed with respect to edges and vertices, i.e. only for the graph structure. If we want to transfer the constraint "Operations possess unique return type" (cf. annotation "1" at edges *return* in $\mathcal{D}_1$ and $\mathcal{D}_2$) to the colimit, we can do that by imposing it to $\mathcal{D}_{13}$, as well, and treat it as extension of the grafical structure. This can be achieved by extending $\mathbb{B}$ to the category $\mathbb{B}'$ =

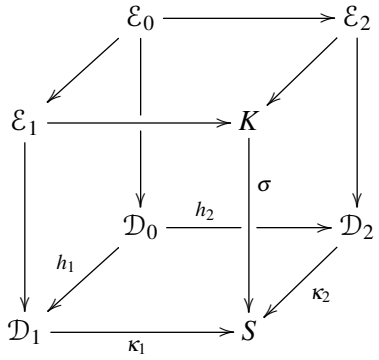


(identities omitted) where $a = s \circ 1$, $b = t \circ 1$. Object U and outgoing arrows depict the uniqueness constraint and its arity shape graph $a \xrightarrow{1} b$, cf. [4, 21]. Objects $\mathcal{D}_i \in Set^{\mathbb{B}'}$ enhance directed multi-graphs by a third set $\mathcal{D}_i(U)$ (of constraint annotations) together with a mapping $\mathcal{D}_i(1) : \mathcal{D}_i(U) \to \mathcal{D}_i(E)$ indicating at which edge in the graph the constraint is annotated. In such a way, the colimit automatically carries all constraints of the component graphs. Since Theorem 8 is valid for arbitrary $\mathbb{B}$, reasoning about the Van Kampen property is possible in the same way as before.

4 Towards Colimits

4.1 Pushouts

The original definition of Van Kampen pushouts was given in [27]: A pushout of a diagram (a span) $\mathcal{D}_1 \xleftarrow{h_1} \mathcal{D}_0 \xrightarrow{h_2} \mathcal{D}_2$ is said to have the Van Kampen property if for any commutative cube



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. It is easy to see that this is an instance of the general definition of Van Kampen colimits in Def.4:

- 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".

In [30] we already stated a characterization of the Van Kampen property for pushouts. It comes in terms of cyclic mapping structures within $\mathcal{D}_0$.

Definition 9 (Domain Cycle, [18]) *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* h_1 *and* h_2*), 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 original criterion for Van Kampen pushouts (Theorem 3 in [30]) came in terms of the absence of arbitrary domain cycles. In order to use the theorem in the present paper we slightly reformulate it: The Van Kampen property can be checked by validating the absence of *proper* domain cycles only:

Theorem 10 (Condition for VK pushout) *A pushout*

$$\begin{array}{ccc} \mathcal{D}_0 & \xrightarrow{h_2} & \mathcal{D}_2 \\ {\scriptstyle h_1}\downarrow & & \downarrow{\scriptstyle \kappa_2} \\ \mathcal{D}_1 & \xrightarrow[\kappa_1]{} & S \end{array}$$

in $\mathbb{G} = Set^{\mathbb{B}}$ *is a VK pushout if and only if there is no proper domain cycle for* h_1 *and* h_2. □

This variant of the theorem easily follows from the original version and Lemma 32 in the appendix.

We see that this already gives a feasible condition to validate the Van Kampen property for pushouts. The plan of the following sections is to transfer this knowledge to special mapping paths in coequalizer diagrams (Sect.4.2) and from there to mapping paths in arbitrary colimits (Sect.4.3).

4.2 From Pushouts to Coequalizers

The following fact is well-known [20]:

Lemma 11 *Let* $B \underset{f}{\overset{g}{\rightrightarrows}} D$ *be two arrows in any category with colimits.*

$$\begin{array}{ccccc} B & \underset{f}{\overset{g}{\rightrightarrows}} & D & \xrightarrow{\kappa_D} & S \\ & {\scriptstyle \kappa_B}\searrow & & \nearrow & \end{array}$$

is a coequalizer diagram if and only if

$$\begin{array}{ccc} B + B & \xrightarrow{[f,g]} & D \\ {\scriptstyle [id,id]}\downarrow & & \downarrow{\scriptstyle \kappa_D} \\ B & \xrightarrow[\kappa_B]{} & S \end{array} \qquad (8)$$

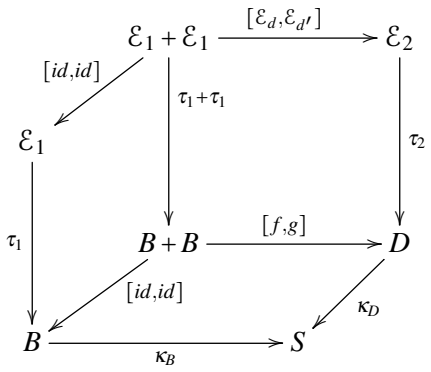
is a pushout. □

The "if"-part has to be made precise: If $h : B \to S, c : D \to S$ is the pushout of $[id, id]$ and $[f, g]$, then c is the coequalizer of f, g and $h = c \circ f = c \circ g$. As a our first new result we show that the equivalence in Lemma 11 extends to the VK property if the underlying category is a presheaf topos $\mathbb{G}$.

Lemma 12 *In $\mathbb{G} = Set^{\mathbb{B}}$ the coequalizer im Lemma 11 has the Van Kampen property, if, and only if the pushout (8) has the Van Kampen property.*

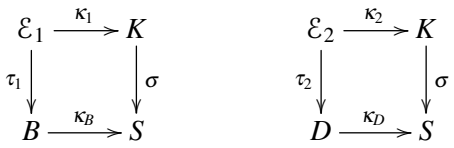
Proof: Let the diagram $\mathcal{D} : \mathbf{2} \to \mathbb{G}$ be defined by $\mathcal{D}_1 = B$, $\mathcal{D}_2 = D$, $\mathcal{D}_d = f$, and $\mathcal{D}_{d'} = g$.

"If-part": Let (8) be a VK pushout and $\tau : \mathcal{E} \Rightarrow \mathcal{D} : \mathbf{2} \to \mathbb{G}$ be a cartesian natural transformation with $\mathcal{D}_1 = B$, $\mathcal{D}_2 = D$, $\mathcal{D}_d = f$, and $\mathcal{D}_{d'} = g$. This yields two pullbacks (see Fact 1) in the rear of



(9)

with the VK pushout in the bottom. Constructing the coequalizer $\mathcal{E}_1 \rightrightarrows \mathcal{E}_2 \xrightarrow{\kappa_2} K$ ($\mathcal{E}_{d'}$, $\mathcal{E}_d$, κ_1), we obtain unique $\sigma : K \to S$ making the resulting cube's front faces



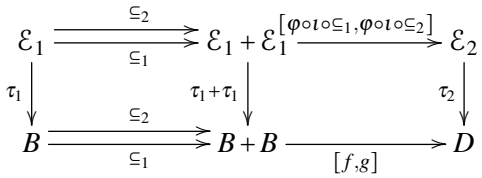
(10)

commute (i.e. $\sigma = \kappa_*(\tau)$). Using Lemma 11 and the VK property of the bottom in (9), the squares in (10) both become pullbacks. This means that $\kappa^*(\kappa_*(\tau)) = \kappa^*(\sigma) = \tau \in \mathbb{G} \Downarrow \mathcal{D}$ up to isomorphism, hence the unit of adjunction $\kappa_* \dashv \kappa^*$ is an isomorphism, i.e. κ^* is an equivalence[8], which yields the desired result by Def.4.

"Only-if-part": Let the coequalizer diagram in Lemma 11 be a VK coequalizer, and let the corresponding pushout of Lemma 11 in the bottom of (9) be accompanied with an arbitrary pullback span in the rear, given by $\tau_1 : \mathcal{E}_1 \to B$, $\tau_2 : \mathcal{E}_2 \to D$, $\varphi : P \to \mathcal{E}_2$, $\psi : P \to \mathcal{E}_1$ and $\mu : P \to B + B$. Then this span must be isomorphic to a span of the form depicted in (9): For the two left faces we get an isomorphism $\iota : \mathcal{E}_1 + \mathcal{E}_1 \to P$, with $\psi \circ \iota = [id, id]$ and $\mu \circ \iota = \tau_1 + \tau_1$, because $\tau_1 + \tau_1$ is a pullback of $[id, id]$ and τ_1 by Fact 1. The composition $\varphi \circ \iota : \mathcal{E}_1 + \mathcal{E}_1 \to \mathcal{E}_2$ establishes a second commutative back square that inherits the pullback property from the given arbitrary back right square via the isomorphism ι since $\mu \circ \iota = \tau_1 + \tau_1$. The composition $\varphi \circ \iota : \mathcal{E}_1 + \mathcal{E}_1 \to \mathcal{E}_2$ can be split into its components $\varphi \circ \iota = [\varphi \circ \iota \circ \subseteq_1, \varphi \circ \iota \circ \subseteq_2]$, where $\subseteq_{1/2}$ are coproduct injections for $\mathcal{E}_1 + \mathcal{E}_1$. Thus we can define a diagram $\mathcal{E} : \mathbf{2} \to \mathbb{G}$ with $\mathcal{E}_{d/d'} := \varphi \circ \iota \circ \subseteq_{1/2}$.

[8] Remember that the counit is always an isomorphism in $\mathbb{G}$, cf. Sect.3.

To see that the given τ_1 and τ_2 provide a cartesian natural transformation, one prolongs the pullback of τ_2 along $[f,g]$ along the two coproduct injections of B into $B+B$:



(that the two left squares are pullbacks follows from extensivity). By pullback composition we obtain $\tau : \mathcal{E} \Rightarrow \mathcal{D} : \mathbf{2} \to \mathbb{G}$.

Constructing the pushout on the top face, we obtain a unique $\sigma : K \to S$ making the squares in (10) commute. Using Lemma 11 for the top face and the VK property of the coequalizer in Lemma 11 we can conclude that the squares in (10) are pullbacks as well. This shows that the bottom of 9 has the VK property. □

In Def.5 we depicted path segments in the form (y, δ, y') where $\delta = d$ or $\delta = d^{op}$ with $d \in \mathbb{I}_1$. In order to simplify notation, we will write (y, p, y') with $p \in \{f, g\}$ in the special case $\mathbb{I} = \mathbf{2}$, i.e. the case of coequalizers $B \overset{g}{\underset{f}{\rightrightarrows}} D$. This means that path segments are equal only if the *names* of their middle component coincide, which is important for the case $f = g$. If $p \in \{f, g\}$ with $-p$ we mean its "complement", i.e. $p = f \Rightarrow -p := g$ and $p = g \Rightarrow -p := f$.

Definition 13 (Path Disjointness) *Let $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be an arbitrary diagram. We say that two different mapping paths P_1 and P_2 in $\mathcal{D}$ of sort X are* disjoint *if non of the path segments in P_1 is weak equal[9] to a path segment in P_2.*

Thus we deliberately excluded disjointness of [ ] and [ ]. This enables simpler formulations for the forthcoming results. Hence in this paper *disjointness implies distinctness*.

Lemma 14 (Domain Cycles and Mapping Paths) *Let $\mathbb{G} = Set^{\mathbb{B}}$ and $X \in \mathbb{B}$. There is a proper domain cycle of sort X for the span $B \xleftarrow{[id,id]} B+B \xrightarrow{[f,g]} D$ if and only if there are $z, z' \in D(X)$ and two disjoint proper mapping paths connecting z and z'.*

Proof: We remind that the notation $[f,g] : B+B \to D$ means that f acts on the first copy of B in $B+B$ and g on the second, e.g. $B_1 = B = B_2$ and $[f,g] : B_1 + B_2 \to D$, $x \in B_1(X)$ for some $X \in \mathbb{B}$, then $[f,g](x) = f(x)$. "⇒". Let $[x_0, x_1, \ldots, x_{2k+1}]$ be a proper domain cycle of elements of $(B+B)(X)$. I.e. we have

$$\begin{array}{rcl} [id,id](x_0) & = & [id,id](x_1) \\ [f,g](x_1) & = & [f,g](x_2) \\ & \vdots & \\ [id,id](x_{2k}) & = & [id,id](x_{2k+1}) \\ [f,g](x_{2k+1}) & = & [f,g](x_0) \end{array}$$

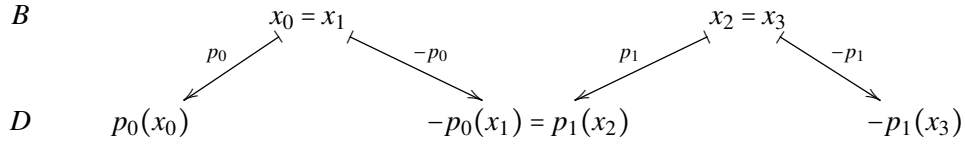
A pair (x_{2i}, x_{2i+1}) in the kernel of $[id,id]$ means that $x_{2i} = x_{2i+1}$ as elements of B, but they occur in different copies of $B+B$ (because the cycle is proper, i.e. $x_{2i} \neq x_{2i+1}$ as elements of $B+B$). Thus, if $p \in \{f,g\}$ is defined on x_{2i}, then $-p$ is defined on x_{2i+1}. A pair (x_{2i+1}, x_{2i+2}) in the kernel of $[f,g]$ means

[9] Recall the definition of weak equality in Def.5.

that $p(x_{2i+1}) = p'(x_{2i+2})$ with $p, p' \in \{f, g\}$ depending on the copy of B, where x_{2i+1}, x_{2i+2} are contained, resp. This establishes a mapping path which starts with the segments

$$(p_0(x_0), p_0^{op}, x_0), (x_1, -p_0, -p_0(x_1)), (p_1(x_2), p_1^{op}, x_2), (x_3, -p_1, -p_1(x_3))$$

where $p_0, p_1 \in \{f, g\}$ according to x_m's membership to a copy of B ($m \in \{0, 1, 2, 3\}$). It can be depicted as follows:



The whole path is the sequence of consecutive pairs

$$[(p_i(x_{2i}), p_i^{op}, x_{2i}), (x_{2i+1}, -p_i, -p_i(x_{2i+1}))]_{0 \le i \le k}$$

where $p_i \in \{f, g\}$ according to x_{2i}'s membership to a copy of B. Because $[f, g](x_{2k+1}) = [f, g](x_0)$ (see Def.9), this non-empty path (its length is $2k + 2 > 0$) connects $p_0(x_0)$ with itself. It remains to show that the constructed path is proper, too. Assume to the contrary that it is not proper, then there are indices $0 \le i < j \le 2k + 1$ with $x_i = x_j$ as elements of B. Since the domain cycle was proper x_i and x_j must occur in different copies of $B + B$. In such a way, the two segments where x_i and x_j, respectively, appear are not weak equal since the corresponding middle components of the two segments are neither equal nor equal up to $(\_)^{op}$, which yields the desired properness.

Since the empty path (being proper and disjoint from the constructed path) also connects $p_0(y_0)$ with itself, the first part of the proof is complete.

"$\Leftarrow$" Assume now that there are $z, z' \in D(X)$ admitting two disjoint proper mapping paths connecting z and z'. Thus both of them must take the form

$$[(z = z_0, p_0^{op}, y_0), (y_0, -p_0, z_1), \ldots, (z_{n-1}, p_{n-1}^{op}, y_{n-1}), (y_{n-1}, -p_{n-1}, z_n)] \tag{11}$$

for some $p_i \in \{f, g\}$. By the remark after Def.5 the first can be concatenated with the reversed second path yielding a cyclic proper (by disjointness) path. Since the two paths were different, the concatenated path is non-empty. We assume that this cyclic path is given by (11), i.e. $z_n = z_0$ and $n \ge 1$.

Define elements of $B + B$ as follows: Let y_i^1 and y_i^2 be in different copies of B, but $y_i^1 = y_i = y_i^2$ as elements of B. The choice of copy depends on the arrangement of f and g: If $p_i = f$, then y_i^1 is in the first copy. If $p_i = g$, then y_i^1 is in the second copy ($i \in \{0, \ldots, n-1\}$). This yields

$$[f, g](y_i^1) = p_i(y_i) \text{ and } [f, g](y_i^2) = -p_i(y_i) \tag{12}$$

We claim that

$$y_0^1, y_0^2, y_1^1, y_1^2, \ldots, y_{n-1}^1, y_{n-1}^2 \tag{13}$$

is a proper domain cycle of $[id, id]$ and $[f, g]$: On the one hand for all $i \in \{0, \ldots, n-1\}$: $[id, id](y_i^1) = [id, id](y_i^2)$. On the other hand, by (12) and the mapping path structure, $[f, g](y_{i-1}^2) = -p_{i-1}(y_{i-1}) = p_i(y_i) = [f, g](y_i^1)$ ($0 < i < n$) and $[f, g](y_{n-1}^2) = -p_{n-1}(y_{n-1}) = z_n = z_0 = p_0(y_0) = [f, g](y_0^1)$. It remains to prove that the domain cycle is proper. Assume for this that there are $0 \le i \le j < n$ such that $y_i^b = y_j^c$ in $(B + B)(X)$ where $b, c \in \{1, 2\}$ and such that $i < j$ or $b \ne c$. This means that $y_i = y_j$ (as elements of B) and

they are in the same copy of B. We can further assume $i < j$, because y_i^1 and y_i^2 are in different copies of B. Consider the four segments

$$(z_i, p_i^{op}, y_i), (y_i, -p_i, z_{i+1}), (z_j, p_j^{op}, y_j), (y_j, -p_j, z_{j+1})$$

of the above mapping path. If $b = c$, then $p_i = p_j$ and the first and the third segments are equal. If $b \neq c$, then $p_i = -p_j$ and the first and the forth segments are weak equal. This contradicts properness of the mapping path and settles the claim. □

Theorem 15 (Condition for VK coequalizers) *Let $\mathbb{G} = Set^{\mathbb{B}}$ and $\overline{\mathcal{D}} : \mathbf{2} \to \mathbb{G}$. The coequalizer diagram*

$$\overline{\mathcal{D}}_1 \underset{\overline{\mathcal{D}}_d}{\overset{\overline{\mathcal{D}}_{d'}}{\rightrightarrows}} \overline{\mathcal{D}}_2 \xrightarrow{\kappa_2} S \qquad \overline{\mathcal{D}}_d \xrightarrow{\kappa_1}$$

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'.

Proof: Let again $B := \overline{\mathcal{D}}_1, D := \overline{\mathcal{D}}_2, f := \overline{\mathcal{D}}_d, g := \overline{\mathcal{D}}_{d'}$.

The coequalizer diagram has the Van Kampen property if and only if the pushout (8) has the Van Kampen property (by Lemma 12) if and only if for all $X \in \mathbb{B}$ there is no proper domain cycle for $[id_B, id_B] : B + B \to B$ and $[f, g] : B + B \to D$ of sort X (by Theorem 10) if and only if for all $X \in \mathbb{B}$ no $z, z' \in D(X)$ can be connected by two disjoint proper mapping paths of sort X (by Lemma 14). □

Recall the already made observations of Example 7: For these simple examples they confirm the statement of Theorem 15.

4.3 From Coequalizers to Colimits

It is well-known [20], that the colimit of $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ can be computed by constructing the coequalizer of

$$\coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)} \underset{\vec{\mathcal{D}}_d}{\overset{\vec{id}}{\rightrightarrows}} \coprod_{j \in \mathbb{I}_0} \mathcal{D}_j \tag{14}$$

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

$$\begin{array}{ccc} \coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)} & \xrightarrow{\vec{\mathcal{D}}_d} & \coprod_{j \in \mathbb{I}_0} \mathcal{D}_j \\ \uparrow & & \uparrow \\ \mathcal{D}_i & \xrightarrow{\mathcal{D}_d} & \mathcal{D}_j \end{array} \qquad \begin{array}{ccc} \coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)} & \xrightarrow{\vec{id}} & \coprod_{i \in \mathbb{I}_0} \mathcal{D}_i \\ \uparrow & & \uparrow \\ \mathcal{D}_i & = & \mathcal{D}_i \end{array}$$

(for all edges $i \xrightarrow{d} j$ in $\mathbb{I}_1$). *We stress the fact that in the left coproduct of (14) an object $\mathcal{D}_i$ occurs as often as there are edges d leaving i in graph $\mathbb{I}$, i.e. if there are e.g. two edges with source i, then two copies of $\mathcal{D}_i$ occur.*

As our second new result we show that the equivalence in (14) extends, for arbitrary topoi, to the VK property.

Lemma 16 *The cocone (1) is VK if and only if the cocone*

$$\coprod_{d\in\mathbb{I}_1}\mathcal{D}_{s(d)} \underset{\vec{\mathcal{D}}_d}{\overset{\vec{id}}{\rightrightarrows}} \coprod_{j\in\mathbb{I}_0}\mathcal{D}_j \xrightarrow{\overline{\kappa}} S \qquad \overline{\kappa}' \tag{15}$$

resulting from constructing the coequalizer in (14) is VK.

Proof: Let $\kappa^* : \mathbb{G}{\downarrow}S \to \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$ be the functor introduced in (3), $\mathbf{2} := 1 \underset{d}{\overset{d'}{\rightrightarrows}} 2$ be the (already used) shape graph for coequalizers, $\overline{\mathcal{D}} : \mathbf{2} \to \mathbb{G}$ the functor mapping this shape graph to the objects and arrows in (14) and

$$\overline{\kappa}^* : \mathbb{G}{\downarrow}S \to \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$$

be the pullback functor for the colimiting cocone in (15). By Definition 4, the lemma is proven, if we can establish an equivalence $\cong$ of categories between $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$ and $\mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$, such that

$$\begin{array}{ccc} & \mathbb{G}{\downarrow}S & \\ {\scriptstyle\kappa^*}\swarrow & & \searrow{\scriptstyle\overline{\kappa}^*} \\ \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} & \xrightarrow[\cong\,?]{} & \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}} \end{array} \tag{16}$$

commutes up to natural isomorphism.

Such an equivalence can be determined as follows: Given $\tau : \mathcal{E} \Rightarrow \mathcal{D}$ in $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$, then - at least in topoi - the pullbacks of τ_i along $id_{\mathcal{D}_i}$ yield a pullback (cf. Fact 1)

$$\begin{array}{ccc} \coprod_{d\in\mathbb{I}_1, s(d)=i}\mathcal{E}_{s(d)} & \xrightarrow{\vec{id}_i^{\,\mathcal{E}}} & \mathcal{E}_i \\ {\scriptstyle\coprod_{d\in\mathbb{I}_1, s(d)=i}\tau_{s(d)}}\downarrow & & \downarrow{\scriptstyle\tau_i} \\ \coprod_{d\in\mathbb{I}_1, s(d)=i}\mathcal{D}_{s(d)} & \xrightarrow{\vec{id}_i^{\,\mathcal{D}}} & \mathcal{D}_i \end{array}$$

for each $i \in \mathbb{I}_0$. These pullbacks can be summed over i yielding the pullback (cf. Fact 2)

$$\begin{array}{ccc} \coprod_{d\in\mathbb{I}_1}\mathcal{E}_{s(d)} & \xrightarrow{\vec{id}^{\,\mathcal{E}}} & \coprod_{i\in\mathbb{I}_0}\mathcal{E}_i \\ {\scriptstyle\coprod_{d\in\mathbb{I}_1}\tau_{s(d)}}\downarrow & & \downarrow{\scriptstyle\coprod_{i\in\mathbb{I}_0}\tau_i} \\ \coprod_{d\in\mathbb{I}_1}\mathcal{D}_{s(d)} & \xrightarrow{\vec{id}^{\,\mathcal{D}}} & \coprod_{i\in\mathbb{I}_0}\mathcal{D}_i \end{array} \tag{17}$$

Moreover for fixed $j \in \mathbb{I}_0$, the pullbacks of τ_j and $\mathcal{D}_d$ yield a pullback

$$\begin{array}{ccc} \coprod_{d\in\mathbb{I}_1, t(d)=j}\mathcal{E}_{s(d)} & \xrightarrow{\vec{\mathcal{E}}_d^{\,j}} & \mathcal{E}_j \\ {\scriptstyle\coprod_{d\in\mathbb{I}_1, t(d)=j}\tau_{s(d)}}\downarrow & & \downarrow{\scriptstyle\tau_j} \\ \coprod_{d\in\mathbb{I}_1, t(d)=j}\mathcal{D}_{s(d)} & \xrightarrow{\vec{\mathcal{D}}_d^{\,j}} & \mathcal{D}_j \end{array}$$

for each $j \in \mathbb{I}_0$ ($\vec{\mathcal{D}}_d{}^j, \vec{\mathcal{E}}_d{}^j$ being appropriate universal coproduct arrows). Again, those pullbacks can be summed over j yielding the second pullback

$$\begin{array}{ccc} \coprod_{d\in\mathbb{I}_1} \mathcal{E}_{s(d)} & \xrightarrow{\vec{\mathcal{E}}_d} & \coprod_{j\in\mathbb{I}_0} \mathcal{E}_j \\ {\scriptstyle \coprod_{d\in\mathbb{I}_1} \tau_{s(d)}}\downarrow & & \downarrow{\scriptstyle \coprod_{j\in\mathbb{I}_0} \tau_j} \\ \coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} & \xrightarrow{\vec{\mathcal{D}}_d} & \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j \end{array} \tag{18}$$

Obviously $\overline{\mathcal{D}} : \mathbf{2} \to \mathbb{G}$ creates the bottom rows in (17, 18). Analogously, let $\overline{\mathcal{E}} : \mathbf{2} \to \mathbb{G}$ form the top rows in (17, 18). Then we have established an assignment $\tau \mapsto (\coprod_{d\in\mathbb{I}_1} \tau_{s(d)}, \coprod_{j\in\mathbb{I}_0} \tau_j)$, the result being a cartesian natural transformation in $\mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$. This assignment

$$\phi : \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D} \to \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}} \tag{19}$$

on objects can be extended to morphisms by similar pullback summation: Let $\tau : \mathcal{E} \Rightarrow \mathcal{D}$, $\tau' : \mathcal{E}' \Rightarrow \mathcal{D}$ and a morphism $\alpha : \tau \Rightarrow \tau'$ be given in $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$. Then – in the same way as before – the summation over the intermediate commutative squares

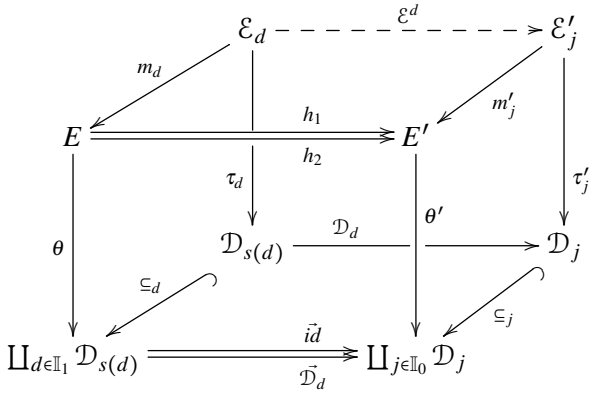
$$\begin{array}{ccc} \coprod_{d\in\mathbb{I}_1, s(d)=i} \mathcal{E}_{s(d)} & \xrightarrow{\vec{id}_i^{\mathcal{E}}} & \mathcal{E}_i \\ {\scriptstyle \coprod_{d\in\mathbb{I}_1, s(d)=i} \alpha_{s(d)}}\downarrow & & \downarrow{\scriptstyle \alpha_i} \\ \coprod_{d\in\mathbb{I}_1, s(d)=i} \mathcal{E}'_{s(d)} & \xrightarrow{\vec{id}_i^{\mathcal{E}'}} & \mathcal{E}'_i \end{array} \qquad \begin{array}{ccc} \coprod_{d\in\mathbb{I}_1, t(d)=j} \mathcal{E}_{s(d)} & \xrightarrow{\vec{\mathcal{E}}_d^{\,j}} & \mathcal{E}_j \\ {\scriptstyle \coprod_{d\in\mathbb{I}_1, t(d)=j} \alpha_{t(d)}}\downarrow & & \downarrow{\scriptstyle \alpha_j} \\ \coprod_{d\in\mathbb{I}_1, t(d)=j} \mathcal{E}'_{s(d)} & \xrightarrow{\vec{\mathcal{E}'}_d^{\,j}} & \mathcal{E}'_j \end{array}$$

yields $\phi(\alpha)_1 = \coprod_{d\in\mathbb{I}_1} \alpha_{s(d)}$ and $\phi(\alpha)_2 = \coprod_{j\in\mathbb{I}_0} \alpha_j$ with $\phi(\alpha) : \phi(\tau) \Rightarrow \phi(\tau')$. Obviously ϕ respects identities and composition, because the coproduct functor has these properties. Thus (19) is a functor.

Recall that $\subseteq^*_-$ depicts the pullback functors along coproduct injections based on chosen pullbacks. For future use, we remark that extensivity yields

$$\subseteq^*_d (\phi(\tau)_1) \cong \tau_{s(d)} \text{ and } \subseteq^*_j (\phi(\tau)_2) \cong \tau_j. \tag{20}$$

It remains to show that ϕ is an equivalence of categories and that (16) commutes up to natural isomorphism. For this consider Fig.3. The front face depicts input data for the hypothetical inverse functor from $\mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$ to $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$, namely $\overline{\mathcal{E}} = E \overset{h_1}{\underset{h_2}{\rightrightarrows}} E'$ (a diagram with shape $\mathbf{2}$) and $(\theta, \theta') : \overline{\mathcal{E}} \Rightarrow \overline{\mathcal{D}}$ a cartesian natural transformation. One constructs pullbacks along coproduct injections: On the left hand side this is performed for all $d \in \mathbb{I}_1$ yielding some $\tau_d : \mathcal{E}_d \to \mathcal{D}_{s(d)}$ for some $\mathcal{E}_d$. *Here, we work with an arbitrary pullback - not necessarily the chosen one. Later, we determine the concrete choice!* On the right hand side, it is performed for all $j \in \mathbb{I}_0$ yielding τ'_j, in this case we take chosen pullbacks.



(21)

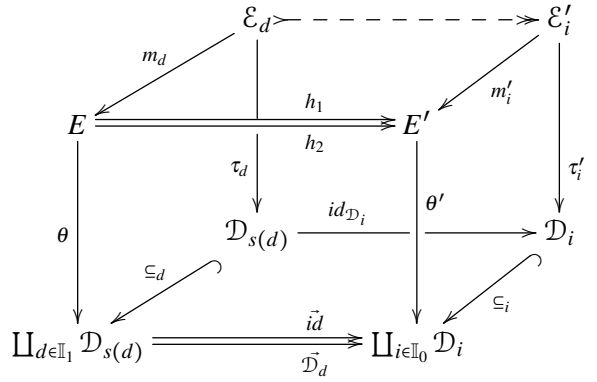
Figure 3: Obtaining ϕ's inverse functor

Note that the bottom square commutes only via $\vec{\mathcal{D}}_d$. By extensivity

$$\theta \cong \coprod_{d\in\mathbb{I}_1} \tau_d \text{ and } \theta' \cong \coprod_{j\in\mathbb{I}_0} \tau'_j \tag{22}$$

There is a unique $\mathcal{E}^d : \mathcal{E}_d \to \mathcal{E}'_j$ (dashed arrow) making the top face with h_2 and the back face commutative. Moreover, the back face is pullback (by pullback composition and decomposition).

For fixed $d \in \mathbb{I}_1$ with $i = s(d)$, we now draw the same diagram with $\mathcal{D}_d$ replaced by id_{D_i} in the bottom of the back face (hence also $\mathcal{D}_j$ replaced by $\mathcal{D}_i$), then the same reasoning via h_1 and $\vec{id}$ yields a different dashed arrow in



which is forced to be an isomorphism by the pullback property of the back face. *Thus our choice of τ_d – the pullback along coproduct injection $\subseteq_d$ – can now concretely be determined:* We can take

$$\tau_d = \tau'_i$$

(hence $\mathcal{E}_d = \mathcal{E}'_i$) for all $i \xrightarrow{d} j \in \mathbb{I}_1$. Substituting this in the back face of (21), we deduce that $\mathcal{E} = (\mathcal{E}^d : \mathcal{E}'_i \to \mathcal{E}'_j)_{d\in\mathbb{I}_1}$ becomes an $\mathbb{I}$-shaped diagram, and $\tau' := (\tau'_i : \mathcal{E}'_i \to \mathcal{D}_i)_{i\in\mathbb{I}_0} : \mathcal{E} \Rightarrow \mathcal{D}$ is a cartesian natural transformation. In this way, we established an assignment $\psi : (\theta, \theta') \mapsto \tau'$ mapping objects of $\mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$ to objects of $\mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$. Functoriality of chosen pullbacks along $\subseteq_i$ extends ψ to a functor.

To show that ϕ and ψ are pseudo-inverses of each other, recall from (22) and the fact that $\tau_d = \tau'_i$ that

$$\psi(\theta, \theta') = \tau' \quad \Rightarrow \quad \subseteq_d^* (\theta) \cong \tau'_i \text{ and } \subseteq_j^* (\theta') = \tau'_j. \tag{23}$$

For any $\tau \in \mathbb{G}^{\mathbb{I}} \Downarrow \mathcal{D}$, let $\tau' := \psi(\phi(\tau))$ then (23) and (20) yield for all $j \in \mathbb{I}_0$

$$\tau'_j \cong \subseteq^*_j (\phi(\tau)_2) \cong \tau_j$$

In the same way for all $(\theta, \theta') \in \mathbb{G}^{\mathbf{2}} \Downarrow \overline{\mathcal{D}}$ and $(\gamma, \gamma') := \phi(\psi(\theta, \theta'))$ with $\tau' := \psi(\theta, \theta')$ we obtain from (22):

$$(\gamma, \gamma') = (\coprod_{d \in \mathbb{I}_1} \tau'_{s(d)}, \coprod_{i \in \mathbb{I}_0} \tau'_i) \cong (\theta, \theta')$$

hence $id \cong \psi \circ \phi$ and $\phi \circ \psi \cong id$, where naturality of isomorphisms arise from universality properties of involved constructions.

Finally, since colimit in (1) and coequalizer in (15) are related via

$$\overline{\kappa} \circ \subseteq_j = \kappa_j$$

for all $j \in \mathbb{I}_0$, pullback composition and the above construction of ψ show that (16) commutes up to natural isomorphism (arising from natural isos due to pseudoriality of pullback functors). □

5 Proof of Theorem 8

The reader may now return to Def.5 in Sect.3 to remember the definition of *mapping paths* within diagram $\mathcal{D} : \mathbb{I} \to \mathbb{G}$. In order to relate them with mapping paths of $\mathbf{2}$-shaped diagrams we define a function φ, which assigns to each proper mapping path in $\mathcal{D}$ a mapping path of diagram

$$\overline{\mathcal{D}} \quad = \quad \coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)} \overset{\vec{id}}{\underset{\vec{\mathcal{D}}_d}{\rightrightarrows}} \coprod_{j \in \mathbb{I}_0} \mathcal{D}_j \ .$$

For this we have to introduce some further notation: In order to visually distinguish mapping paths in $\mathcal{D}$ from those in $\overline{\mathcal{D}}$, a path segment of a mapping path in $\mathcal{D}$ will be depicted $(y, \delta, y')^{\mathcal{D}}$ whereas $(z, h, z')^{\overline{\mathcal{D}}}$ depicts a segment of paths in $\overline{\mathcal{D}}$. For simplicity, we let $h \in \{\vec{id}, \vec{\mathcal{D}}_d\}$. This means that two segments $(\_, h, \_)^{\overline{\mathcal{D}}}$ and $(\_, h', \_)^{\overline{\mathcal{D}}}$ are distinct if the *names* h and h' are different, although probably $\vec{id} = \vec{\mathcal{D}}_d$.

We already claimed that for each $X \in \mathbb{B}$ all elements of all sets $(\mathcal{D}_i(X))_{i \in \mathbb{I}_0}$ are disjoint. But now we also have to make sure that the carriers of $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ and $\coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)}$ are sortwise disjoint (in the diagram $\overline{\mathcal{D}}$). This can be achieved for any $i \xrightarrow{d} j$ in $\mathbb{I}_1$ by making the coproduct injection

$$\mathcal{D}_i \overset{\subseteq^i_d}{\rightarrowtail} \coprod_{d \in \mathbb{I}_1, s(d) = i} \mathcal{D}_{s(d)}$$

a renaming operator: For $y \in \mathcal{D}_i$ let $(y, d) := \subseteq^i_d (y)$, thus we amend all elements of $\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ by adding the index of the set they belong to in $\coprod_{d \in \mathbb{I}_1} \mathcal{D}_{s(d)}$. This yields

$$\vec{id}(y, d) = y \text{ and } \vec{\mathcal{D}}_d(y, d) = \mathcal{D}_d(y).$$

We now define an assignment of one $\mathcal{D}$-segment to two $\overline{\mathcal{D}}$-segments and let φ be the natural extension to segment sequences of proper mapping paths. The segment assignment is

$$\begin{aligned} (y, d, y')^{\mathcal{D}} \mapsto \quad & (y, \vec{id}^{op}, (y, d))^{\overline{\mathcal{D}}}, ((y, d), \vec{\mathcal{D}}_d, y')^{\overline{\mathcal{D}}}, \\ (y', d^{op}, y)^{\mathcal{D}} \mapsto \quad & (y', \vec{\mathcal{D}}_d{}^{op}, (y, d))^{\overline{\mathcal{D}}}, ((y, d), \vec{id}, y)^{\overline{\mathcal{D}}}. \end{aligned}$$

We claim the following 4 properties:

1. φ is a mapping from the set $\wp$ of proper paths in $\mathcal{D}$ to the set $\overline{\wp}$ of proper paths in $\overline{\mathcal{D}}$ which connect elements of $\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i$.
2. $\varphi:\wp\to\overline{\wp}$ is bijective.
3. P_1 and P_2 are disjoint if and only if $\varphi(P_1)$ and $\varphi(P_2)$ are disjoint.
4. For all $z,z'\in\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i$: P connects z,z' in $\mathcal{D}$, if and only if $\varphi(P)$ connects z,z' in $\overline{\mathcal{D}}$.

Property 1 is easy to see, because if $\varphi(P)$ would not be proper, it would contain two weak equal segments S_1 and S_2. Their middle component is equal up to $(\_)^{op}$, such that they can not be the image of a single $\mathcal{D}$-segment. We only consider the example of two weak equal segments $S_1=(z,\vec{\mathcal{D}}_d,z')^{\overline{\mathcal{D}}}$, $S_2=(z',\vec{\mathcal{D}}_d{}^{op},z)^{\overline{\mathcal{D}}}$, all other cases are similar. In this case we must have $z=(y,d)$ and $z'=\mathcal{D}_d(y)$ for some $d:i\to j$ and $y\in\mathcal{D}_i$. Thus S_1 is in the image of segment $(y,d,z')^{\mathcal{D}}$ and S_2 is in the image of weak equal segment $(z',d^{op},y)^{\mathcal{D}}$, which is impossible, because P is proper.

Property 2: φ is injective, because two different paths in $\mathcal{D}$ possess a first index at which their segments differ. Let $S_1=(y_1,\delta_1,y_1')$ and $S_2=(y_2,\delta_2,y_2')$ be these segments. Their images differ according to the above assignment (at the last component of the first and the first component of the second segment), if $\delta_1\neq\delta_2$. Hence we can assume w.l.o.g. $\delta_1=d=\delta_2$ for some d. Their images coincide if $y_1=y_2$, but then $y_1'=\mathcal{D}_d(y_1)=\mathcal{D}_d(y_2)=y_2'$ and thus $S_1=S_2$. It remains to prove surjectivity: Let $\overline{P}$ be a proper path in $\overline{\mathcal{D}}$ connecting two elements of $\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i$. Obviously its length is $2n$ for some $n\geq 0$. We prove existence of a preimage by induction over n. $n=0$: The empty path has preimage the empty path. Induction step: We must have

$$\overline{P}=[(z,h^{op},z')^{\overline{\mathcal{D}}},(z',h',z'')^{\overline{\mathcal{D}}},\overline{P}']$$

for some path $\overline{P}'$ of length $2n\geq 0$ and with $h,h'\in\{\vec{id},\vec{\mathcal{D}}_d\}$. By the above amendment, we must have $z'=(y,d)$ for some $d:i\to j\in\mathbb{I}_1$ and $y\in\mathcal{D}_i$. If $h=h'$, then $\overline{P}$ would not be proper, thus $h=\vec{id}$ and $h'=\vec{\mathcal{D}}_d$ or vice versa. In the first case its preimage segment is $S=(y,d,\mathcal{D}_d(y))^{\mathcal{D}}$, in the second case it is $S=(\mathcal{D}_d(y),d^{op},y)^{\mathcal{D}}$. Path $\overline{P}$ thus has preimage $[S,P']$ with $\varphi(P')=\overline{P}'$ (by induction hypotheses).

Property 3 can be shown similar to injectivity now taking into account weak equal segments and *property 4* is obvious.

This behaviour of φ yields

Lemma 17 *Let $\mathcal{D}:\mathbb{I}\to\mathbb{G}$ be a diagram 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'.*
- $\forall z,z'\in\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i(X)$*: There are no two disjoint proper mapping paths in $\overline{\mathcal{D}}$ connecting z and z'.*

□

Collecting all previous considerations, we obtain

Theorem 18 (Characterization of VK cocones with disjoint paths) *Let $\mathbb{G}=Set^{\mathbb{B}}$ be a presheaf topos and $\mathcal{D}:\mathbb{I}\to\mathbb{G}$ be a diagram with $\mathbb{I}$ a finite directed multigraph. 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 disjoint proper mapping paths in $\mathcal{D}$ connecting z and z'.

Proof:

$\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$ has the Van Kampen property

Lemma 16 $\Longleftrightarrow$ $\overline{\mathcal{D}} \overset{\overline{\kappa}}{\Rightarrow} \Delta S$ i.e. $\coprod_{d\in\mathbb{I}_1} \mathcal{D}_{s(d)} \underset{\overset{\vec{\mathcal{D}}_d}{\overset{\vec{id}}{\rightrightarrows}}}{\overset{\overline{\kappa}'}{\longrightarrow}} \coprod_{j\in\mathbb{I}_0} \mathcal{D}_j \xrightarrow{\overline{\kappa}} S$ has the Van Kampen property

Thm. 15 $\Longleftrightarrow$ $\forall X \in \mathbb{B} : \forall z, z' \in \coprod_{i\in\mathbb{I}_0} \mathcal{D}_i(X) :$
There are no two disjoint proper paths in $\overline{\mathcal{D}}$ from z to z'

Lemma 17 $\Longleftrightarrow$ $\forall X \in \mathbb{B} : \forall i, j \in \mathbb{I}_0 : \forall z \in \mathcal{D}_i(X), \forall z' \in \mathcal{D}_j(X) :$
There are no two disjoint proper paths in $\mathcal{D}$ from z to z'

□

Proof of Theorem 8: In order to prove Theorem 8 we need to get rid of disjointness in Thm.18. The "Enforcing-Disjointness-Lemma" 30 in the appendix shows that the existence of two different proper paths from z_1 to z_1' entails existence of two *disjoint* proper paths from possibly different z_2 to z_2'.
"⇐": Because we stipulated that disjointness implies distinctness (see Def.13 and the remarks added thereafter), this follows from Theorem 18.
"⇒": This follows from the "Enforcing-Disjointness-Lemma" and Theorem 18. □

6 Practical Guidelines

A main use case of the previous results can be found in software engineering and especially in model-driven software designs: Components of diagrams are models (e.g. data models or metamodels which govern the admissible structure of models), morphisms are relations between the models. As described in the introduction, model assembly is often important (cf. Sect.3.3). It shall not only be carried out on a syntactical level (models), but in the same way on the semantical level (instances) such that assembled instances can correctly be decomposed into their original instances by pullback.

It is a goal to efficiently verify whether *compositionality* holds, i.e. whether the Van Kampen (henceforth abbreviated "VK") property is satisfied. Since model composition always requires computation of colimits, VK-verification at the same time is desirable. In this section we will describe (1) a more efficient colimit computation compared to Fact 6 and (2) how to simultaneously check the VK-property.

To further reduce verification effort, we will first look for criteria to decide, for a given diagram, if VK holds or not, without checking explicitly the path conditions of Theorem 8. Moreover, we will show that – in cases where the path condition is needed – one does not need to check the condition for *all* $i, j \in \mathbb{I}_0$ but only for a smaller subset of indices.

6.1 Relevant Types of Mapping Paths

We will discuss now what kinds of paths and what pairs of paths we really need to check in practice. The attentive reader may have noticed already that properness excludes cycles w.r.t. path segments but does not exclude cycles w.r.t. elements. Let $P = [(y_0, \delta_0, y_1), (y_1, \delta_1, y_2), \ldots, (y_{n-1}, \delta_{n-1}, y_n)]$ be a mapping path in a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ with finite $\mathbb{I}$. By ι_i, we denote the unique vertex in $\mathbb{I}_0$ with $y_i \in \mathcal{D}_{\iota_i}$.

Definition 19 *A mapping path is called* inner-cycle free*, if for all indices* $0 \le i < j \le n$ *with* $j - i \le n - 1$*:* $y_i \neq y_j$.

Empty mapping paths or paths of length 1 are inner-cycle free by definition. Note, that we allow P to be an "outer" cycle, i.e., $y_0 = y_n$. Lemma 31 in the appendix shows that any path can be made inner-cycle free. Obviously, this reduction preserves properness and disjointness of mapping paths. Thus we obtain the following corollary of Theorems 8 and 18:

Corollary 20 *Let $\mathbb{G} = Set^{\mathbb{B}}$ be a presheaf topos and $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be a diagram with $\mathbb{I}$ a finite directed multigraph. Let*

$$\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$$

be a colimiting cocone. The following are equivalent:

(1) *The cocone is a Van Kampen cocone*
(2) $\forall X \in \mathbb{B}, i, j \in \mathbb{I}_0, z \in \mathcal{D}_i(X), z' \in \mathcal{D}_j(X)$: *There are no two different proper paths from z to z'*
(3) $\forall X \in \mathbb{B}, i, j \in \mathbb{I}_0, z \in \mathcal{D}_i(X), z' \in \mathcal{D}_j(X)$: *There are no two disjoint proper paths from z to z'*
(4) $\forall X \in \mathbb{B}, i, j \in \mathbb{I}_0, z \in \mathcal{D}_i(X), z' \in \mathcal{D}_j(X)$: *There are no two disjoint inner-cycle free proper paths from z to z'* □

In addition to a better variety of VK characterisations, this result also provides a high degree of freedom for the implementation of algorithms: The result does not depend on whether an algorithm generates only disjoint pairs of paths, or also builds paths with inner cycles. However, every algorithm based on Corollary 20 still has to traverse all components $(\mathcal{D}_i)_{i \in \mathbb{I}_0}$. The next sections will explain why the traversal of components can significantly be reduced.

6.2 Cyclic Shape Graphs

A *directed cycle* in $\mathbb{I}$ is a set of pairwise distinct edges $d_0, \ldots, d_{n-1}$ in $\mathbb{I}_1$ for some $n \geq 1$ with $t(d_{i-1}) = s(d_{i \bmod n})$ $(i \in \{1, \ldots, n\})$. If $n = 1$, the cycle is also called a loop (cf. Example 7). Let $ends(d) = \{s(d), t(d)\}$ be the set of endpoints of an edge d, then an *undirected cycle* in $\mathbb{I}$ is a set of pairwise distinct edges $d_0, \ldots, d_{n-1}$ in $\mathbb{I}_1$ for some $n \geq 2$ with $ends(d_{i-1}) \cap ends(d_{i \bmod n}) \neq \varnothing$ $(i \in \{1, \ldots, n\})$. An example for an undirected cycle is the shape graph $\mathbf{2}$ and also the shape graph in the example of Sect.3.3.

An important observation is that VK is violated, if $\mathbb{I}$ posseses a directed cycle

$$p = (i_0 \overset{d_0}{\to} i_1 \overset{d_1}{\to} \ldots \overset{d_{n-1}}{\to} i_n = i_0)$$

and if for some $X \in \mathbb{B}$ and some $i \in \{i_0, \ldots, i_n\}$ the component $\mathcal{D}_i(X)$ is a finite set. To see this, let w.l.o.g. $i = i_0$ and

$$\mathcal{D}_p := \mathcal{D}_{d_{n-1}} \circ \ldots \circ \mathcal{D}_{d_1} \circ \mathcal{D}_{d_0} : \mathcal{D}_{i_0}(X) \to \mathcal{D}_{i_n}(X) = \mathcal{D}_{i_0}(X).$$

be the corresponding composed function. Obviously, there exist $y \in \mathcal{D}_{i_0}(X)$ and $1 \leq k \leq |\mathcal{D}_{i_0}(X)|$ such that $\mathcal{D}_p^k(y) = y$, where the mappings can be chosen such that this results in a non-empty proper mapping path connecting y with itself: Because the empty path also connects y with itself (cf. Def.5), VK is violated by Cor.20.

6.3 Specialized Construction of Colimits

In this section, we prepare a general and more efficient algorithm for VK verification, which runs in the background of a colimit computation. For this we need to distinguish between different characteristics of shape graph $\mathbb{I}$ and derive from that a specialized colimit construction.

The universal construction recipe for colimits: (14) shows how to construct the colimit of any diagram in a uniform way by means of a coequalizer. It allows to prove general results about colimits (as we have demonstrated in the previous sections). Especially, in case of graphs $\mathbb{I}$ with infinite descending chains and/or with directed cycles, this universal recipe is the best we have. E.g. colimit construction of

$$\mathcal{D}_1 \underset{\mathcal{D}_{d'}}{\overset{\mathcal{D}_d}{\rightrightarrows}} \mathcal{D}_2,$$

and its VK verification is based on mapping paths within the coproduct $\mathcal{D}_1 + \mathcal{D}_2$ and there is no way of minimizing the space for the investigation.

Towards a simplified colimit computation and VK verification Coequalizers, however, can be investigated within a smaller space: It is not VK, if we can find for the corresponding diagram $\mathcal{D} : \mathbf{2} \to Set^{\mathbb{B}}$ a sort $X \in \mathbb{B}$ and an element $y \in \mathcal{D}_1(X)$ such that $\mathcal{D}_d(y) = \mathcal{D}_{d'}(y)$, thus reducing investigations to $\mathcal{D}_1$. However, even if $\mathcal{D}_d$ and $\mathcal{D}_{d'}$ do have sortwise disjoint images, VK may be violated. Note, that those image disjoint diagrams are exactly the diagrams we obtain when constructing pushouts, in the traditional way, by means of sums and coequalizer. To have VK we can require, in addition, that $\mathcal{D}_d$ and $\mathcal{D}_{d'}$ are monic, since then $[\mathcal{D}_d, \mathcal{D}_{d'}]$ is monic and hence the pushout along this mono is VK (because each topos is adhesive [17, 27]). In these special cases, this provides again significant simplification.

Even in the cases left unclear, the check of the VK property for coequalizers must not utilize the condition in Cor.20, which is based on the universal construction recipe (14) for colimits[10]. Instead, we can use directly the condition in Theorem 15, i.e. we need to check the condition in Cor.20 only for the case $i = j = 2$, thus reducing investigations to $\mathcal{D}_2$. We will see in the forthcoming parts that all these effects can often be used in more general cases to reduce analysis effort.

Beside these effects, there may be components that do not contribute to the construction of the colimit at all. In many cases, this simplifies colimit construction, because it is not necessary to compute a quotient of the *entire* coproduct $\coprod_{j \in \mathbb{I}_0} \mathcal{D}_j$. Moreover, we will investigate how certain further assumptions on the properties of arrows in $\mathcal{D}$ (image-disjointness, injectivity) also simplify the algorithm. We will discuss some further examples to motivate the announced specialized and practical construction of colimits.

Since the case of directed cycles can immediately be handled in practical situations[11], we assume from now on that $\mathbb{I}$ *is finite and has no directed cycles*.

Irrelevant components For all indices in $\mathbb{I}$ with no incoming and exactly one outgoing edge, the corresponding component does not contribute to the construction of the colimit. Typical examples are

$$\mathcal{D}_1 \xrightarrow{\mathcal{D}_d} \mathcal{D}_2 \qquad\qquad \mathcal{D}_1 \xrightarrow{\mathcal{D}_{d_1}} \mathcal{D}_3 \xleftarrow{\mathcal{D}_{d_2}} \mathcal{D}_2,$$

(in an arbitrary category). For the left diagram $\mathcal{D}_2$ can be taken as the colimit object and we can set $\kappa_2 := id_{\mathcal{D}_2}$, $\kappa_1 := \mathcal{D}_d$. For the right diagram $\mathcal{D}_3$ can serve as colimit object and we have $\kappa_3 := id_{\mathcal{D}_3}$, $\kappa_1 := \mathcal{D}_{d_1}$, $\kappa_2 := \mathcal{D}_{d_2}$.

[10] Note, that we could apply the universal construction recipe again to the diagram in (14) and so on.

[11] ... where, presumably, components are finite artefacts ...

Jump-over components For all indices in $\mathbb{I}$ with exactly one incoming and one outgoing edge we can jump over the corresponding component. As examples, we consider the diagrams

$$\mathcal{D}_0 \xrightarrow{\mathcal{D}_{d_1}} \mathcal{D}_1 \xrightarrow{\mathcal{D}_{d_2}} \mathcal{D}_2 \qquad\qquad \mathcal{D}_3 \xleftarrow{\mathcal{D}_{d_3}} \mathcal{D}_0 \xrightarrow{\mathcal{D}_{d_1}} \mathcal{D}_1 \xrightarrow{\mathcal{D}_{d_2}} \mathcal{D}_2,$$

(in an arbitrary category). For the left diagram $\mathcal{D}_2$ can be taken as the colimit object and we have $\kappa_2 := id_{\mathcal{D}_2}$, $\kappa_1 := \mathcal{D}_{d_2}$, $\kappa_0 := \mathcal{D}_{d_2} \circ \mathcal{D}_{d_1}$. For the right diagram the colimit object is obtained by the pushout of $\mathcal{D}_{d_2} \circ \mathcal{D}_{d_1} : \mathcal{D}_0 \to \mathcal{D}_2$ and $\mathcal{D}_{d_3} : \mathcal{D}_0 \to \mathcal{D}_3$. The missing injections are given by $\kappa_1 := \kappa_2 \circ \mathcal{D}_{d_2}$ and $\kappa_0 := \kappa_2 \circ \mathcal{D}_{d_2} \circ \mathcal{D}_{d_1} (= \kappa_3 \circ \mathcal{D}_{d_3})$.

Minimal components We consider diagrams for coequalizer and pushouts, respectively,

$$\mathcal{D}_1 \underset{\mathcal{D}_{d'}}{\overset{\mathcal{D}_d}{\rightrightarrows}} \mathcal{D}_2 \qquad\qquad \mathcal{D}_1 \xleftarrow{\mathcal{D}_d} \mathcal{D}_0 \xrightarrow{\mathcal{D}_{d'}} \mathcal{D}_2$$

and the corresponding diagrams according to the universal recipe (14)

$$\mathcal{D}_1 + \mathcal{D}_1 \underset{[\subseteq_2 \circ \mathcal{D}_d, \subseteq_2 \circ \mathcal{D}_{d'}]}{\overset{[\subseteq_1, \subseteq_1]}{\rightrightarrows}} \mathcal{D}_1 + \mathcal{D}_2 \qquad\qquad \mathcal{D}_0 + \mathcal{D}_0 \underset{[\subseteq_1 \circ \mathcal{D}_d, \subseteq_2 \circ \mathcal{D}_{d'}]}{\overset{[\subseteq_0, \subseteq_0]}{\rightrightarrows}} \mathcal{D}_0 + \mathcal{D}_1 + \mathcal{D}_2$$

In case of coequalizer we construct, usually, a quotient of $\mathcal{D}_2$ and not of $\mathcal{D}_1 + \mathcal{D}_2$ and, in case of pushouts we construct a quotient of $\mathcal{D}_1 + \mathcal{D}_2$ and not of $\mathcal{D}_0 + \mathcal{D}_1 + \mathcal{D}_2$. Also in the example in Figure 1 we factorize the sum $\mathcal{D}_1 + \mathcal{D}_2 + \mathcal{D}_3$ and not the sum of all 6 components. In all three cases we build first the coproduct of all minimal components and construct then a quotient of this restricted coproduct.

Definition 21 (Minimal index) *For a finite directed multigraph $\mathbb{I}$ we denote by $Min(\mathbb{I})$ the set of all* (local) minimal indices*, i.e., of all vertices in $\mathbb{I}_0$ without outgoing edges. For a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ we say that $\mathcal{D}_i$ is a* minimal component *if $i \in Min(\mathbb{I})$.*

Since $\mathbb{I}$ is finite and has no directed cycles, each index is either minimal or there exists a non-empty finite sequence of edges to at least one minimal index. This fact will be used several times in the sequel. To construct the colimit of a diagram with finite $\mathbb{I}$ with no directed cycles we need, in practice, only the minimal components as outlined below.

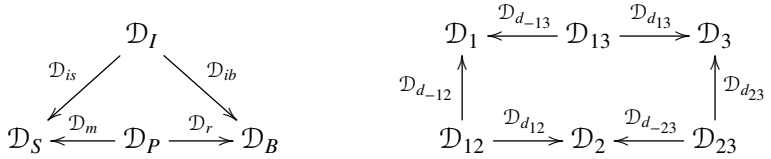
Branching components The essence in constructing the colimit of a diagram is to converge diverging branches in the diagram (in a minimal way). In presheaf topoi this can be done by sortwise identifying certain elements. What elements, however, have to be identified?

In case of coequalizers we have to identify for all sorts $X \in \mathbb{B}$ and all $y \in \mathcal{D}_1(X)$ the two elements $\mathcal{D}_d(y)$ and $\mathcal{D}_{d'}(y)$ in $\mathcal{D}_2(X)$. These primary identifications induce further identifications when we construct the smallest congruence in $\mathcal{D}_2$ comprising all these primary identifications. For pushouts we have to identify for all $X \in \mathbb{B}$ and all $y \in \mathcal{D}_0(X)$ the two elements $\mathcal{D}_d(y)$ and $\mathcal{D}_{d'}(y)$ seen as elements in $\mathcal{D}_1(X) + \mathcal{D}_2(X)$. In this case, we construct then the smallest congruence in $\mathcal{D}_1 + \mathcal{D}_2$ comprising all the primary identifications.

Now we look at slightly more general diagrams. First, we consider three parallel arrows.

$$\mathcal{D}_1 \underset{\mathcal{D}_{d_3}}{\overset{\mathcal{D}_{d_1}}{\overset{\mathcal{D}_{d_2}}{\rightrightarrows}}} \mathcal{D}_2$$

In this case, we have to identify for all sorts $X \in \mathbb{B}$ and all $y \in \mathcal{D}_1(X)$ the three elements $\mathcal{D}_{d_1}(y)$, $\mathcal{D}_{d_2}(y)$ and $\mathcal{D}_{d_3}(y)$ in $\mathcal{D}_2(X)$, and then we construct the smallest congruence in $\mathcal{D}_2$ comprising these identifications. Second, we consider two generalizations of pushouts



A situation, as in the left diagram, appears, for example, if we want to avoid that the instantiation of a "parameterized specification" $\mathcal{D}_P \xrightarrow{\mathcal{D}_r} \mathcal{D}_B$ via a "match" $\mathcal{D}_P \xrightarrow{\mathcal{D}_m} \mathcal{D}_S$ generates two copies of a specification $\mathcal{D}_I$ that had been imported as well by the "body" $\mathcal{D}_B$ of the parameterized specification as by the "actual parameter" $\mathcal{D}_S$. The diagram on the right is taken from the example in Figure 1.

In the left diagram we have to identify for all $X \in \mathbb{B}$ and all $y \in \mathcal{D}_P(X)$ the two elements $\mathcal{D}_m(y)$ and $\mathcal{D}_r(y)$ seen as elements in $\mathcal{D}_S(X) + \mathcal{D}_B(X)$. In addition, we have to identify for all $z \in \mathcal{D}_I(X)$ the two elements $\mathcal{D}_{is}(z)$ and $\mathcal{D}_{ib}(z)$, again seen as elements in $\mathcal{D}_S(X) + \mathcal{D}_B(X)$.

In the right diagram, we have, analogously, that the elements in $\mathcal{D}_{12}$ force identifications of elements in $\mathcal{D}_1$ and $\mathcal{D}_2$, seen as elements in $\mathcal{D}_1 + \mathcal{D}_2 + \mathcal{D}_3$, the elements $\mathcal{D}_{13}$ force identifications of elements in $\mathcal{D}_1$ and $\mathcal{D}_3$, seen as elements in $\mathcal{D}_1 + \mathcal{D}_2 + \mathcal{D}_3$, and the elements in $\mathcal{D}_{23}$ force identifications of elements in $\mathcal{D}_2$ and $\mathcal{D}_3$, seen as elements in $\mathcal{D}_1 + \mathcal{D}_2 + \mathcal{D}_3$.

Generalizing the examples we want to coin the following definition.

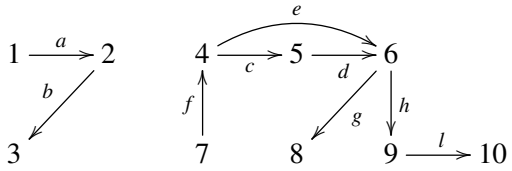
Definition 22 (Branching index) *For a finite directed multigraph $\mathbb{I}$ we denote by $Br(\mathbb{I})$ the set of all* branching indices*, i.e., of all indices with, at least, two outgoing edges. For a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ we say that $\mathcal{D}_i$ is a* branching component *if $i \in Br(\mathbb{I})$.*

A sequence of edges $p = (i_0 \xrightarrow{d_0} i_1 \xrightarrow{d_1} \ldots \xrightarrow{d_{n-1}} i_n)$ in $\mathbb{I}$ is called a branch*, if $i_0 \in Br(\mathbb{I})$ and $i_n \in Min(\mathbb{I})$.*

Note, that $Br(\mathbb{I})$ and $Min(\mathbb{I})$ are disjoint by definition. Thus any branch has at least length 1. Note further, that branching indices can be connected in $\mathbb{I}$, in contrast to minimal indices.

To illustrate our discussion and definitions we consider a simple toy example.

Example 23 *Let $\mathbb{I}$ =*



Here we have $Min(\mathbb{I}) = \{3, 8, 10\}$, $Br(\mathbb{I}) = \{4, 6\}$ and the four branches $(4 \xrightarrow{c} 5 \xrightarrow{d} 6 \xrightarrow{g} 8)$, $(4 \xrightarrow{c} 5 \xrightarrow{d} 6 \xrightarrow{h} 9 \xrightarrow{l} 10)$, $(4 \xrightarrow{e} 6 \xrightarrow{g} 8)$, and $(4 \xrightarrow{e} 6 \xrightarrow{h} 9 \xrightarrow{l} 10)$.

The indices 1 and 7 are irrelevant and we can jump over the indices 5 and 9. We can also jump over the index 2, but since 1 is irrelevant also 2 becomes irrelevant. Be aware that the edges c, d, e constitute an undirected cycle in $\mathbb{I}$.

Branching indices give rise to special positions in mapping paths:

Definition 24 (Branching Position in Proper Paths) *For a pair of subsequent segments*

$$(y_{j-1}, d^{op}, y_j), (y_j, d', y_{j+1})$$

of a proper mapping path P (hence $d \neq d'$) the position j is called a branching position *of P. Consequently ι_j is a branching index of $\mathbb{I}$.*

A specialized construction of colimits Now, we have everything at hand to describe a construction of colimits in presheaf topoi that specializes the construction outlined in Fact 6. Let $\mathbb{I}$ be a finite directed multigraph with no directed cycles. Then we can construct the colimit of a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ as follows:

1. Construct the coproduct $\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i$ (by sortwise coproducts in *Set*).

2. For each pair $p = (i \xrightarrow{d} \ldots \to j)$, $p' = (i \xrightarrow{d'} \ldots \to j')$ of branches in $\mathbb{I}$ with common source $i \in Br(\mathbb{I})$ and $d \neq d'$, and for each sort $X \in \mathbb{B}$ there is the set

$$\approx_X^{p,p'} := \{(\subseteq_j (\mathcal{D}_p(y)), \subseteq_{j'} (\mathcal{D}_{p'}(y))) \mid y \in \mathcal{D}_i(X)\}$$

   of pairs (primary identifications) in $\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i$. Each pair is represented by a primary mapping path connecting the pair's components[12]. By $\approx_X$ we denote the union of all those sets $\approx_X^{p,p'}$ for the sort X. This gives us a family $\approx = (\approx_X)_{X \in \mathbb{B}}$ of binary relations in $\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i$.

3. Construct the smallest congruence $\cong = (\cong_X)_{X \in \mathbb{B}}$ in $\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i$ which comprises $\approx = (\approx_X)_{X \in \mathbb{B}}$ by enlargement with transitive (i.e. concatenation of primary mapping paths) and reflexive (empty mapping paths) closure[13][14].

4. Construct the colimit object as the sortwise quotient $(\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i)/\cong$ and get, in such a way, also the canonical morphisms $[\,]_\cong : \coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i \to (\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i)/\cong$.

5. The colimiting cocone of diagram $\mathcal{D}$ is given by

$$\mathcal{D} \overset{\kappa}{\Rightarrow} (\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i)/\cong \tag{24}$$

   where $\kappa_i := [\,]_\cong \circ \subseteq_i$ for all minimal indices $i \in Min(\mathbb{I})$ and $\kappa_i := \kappa_j \circ \mathcal{D}_p$ for all other indices $i \in \mathbb{I}_0 \setminus Min(\mathbb{I})$, where $p = (i \to \ldots \to j)$ is an edge sequence from i to $j \in Min(\mathbb{I})$. See below for a detailed explanation. Note, that the definition of κ_i is independent of the choice of p since we have, by construction, $\kappa_j \circ \mathcal{D}_p = \kappa_{j'} \circ \mathcal{D}_{p'}$ for all branches $p = (i \to \ldots \to j)$, $p' = (i \to \ldots \to j')$ in $\mathbb{I}$ with a common source.

The main advantage of this construction is that mapping paths are now computed traversing branching components only. These components, however, are often just tiny "connectors" as in Fig.1.

Proof of (24): Let $\subseteq : \coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i \to \coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$ be the embedding morphism. Recall from Fact 6, (2) that $z \equiv^p z'$, if there is a proper mapping path from z to z'. In order to distinguish ordinary mapping paths in $\mathcal{D}$ (according to Def.5) from special paths arising from steps 2 and 3 above, we call the latter *branching mapping paths* in this proof.

It is a well-known fact that there is a unique $Set^{\mathbb{B}}$-homomorphism f making the diagram

$$\begin{array}{ccc} \coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i & \xrightarrow{\quad [\,]_\cong \quad} & (\coprod_{i \in Min(\mathbb{I})} \mathcal{D}_i)/\cong \\ {\scriptstyle [\,]_\equiv \circ \subseteq}\downarrow & & \downarrow{\scriptstyle f} \\ (\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)/\equiv & \xrightarrow[\quad h \quad]{} & (\coprod_{i \in \mathbb{I}_0} \mathcal{D}_i)/\equiv^p \end{array}$$

[12] It can be shown inductively over the path length that each pair can even be represented by a path *with exactly one branching position*.

[13] $\approx$ is already symmetric by definition and the transitive closure preserves symmetry.

[14] It is easy to see that compatibility with operation symbols holds.

commutative, because $[]_{\cong}$ is an epimorphism and its kernel is contained in the kernel of $[]_{\equiv} \circ \subseteq$ (because each branching path is a mapping path). The existence of isomorphism h follows from Fact 6. It is easy see from the path reduction procedure (Lemma 29), that $h([y]_{\equiv}) = [y]_{\equiv^p}$ for all y. To prove (24), we will now show that f is bijective.

It is easy to see that f is surjective: Let $[z]_{\equiv^p}$ be an element in the codomain of f, $z \in \mathcal{D}_i(X)$ and p be a (possibly empty) edge sequence in $\mathbb{I}$ from i to a minimal index j. Commutativity of the above square yields for all $y \in \mathcal{D}_j(X)$:

$$f([y]_{\cong}) = [y]_{\equiv^p} \tag{25}$$

Hence $f([\mathcal{D}_p(z)]_{\cong}) = [\mathcal{D}_p(z)]_{\equiv^p}$ which is equal to $[z]_{\equiv^p}$ because the edge sequence yields a proper path from z to $\mathcal{D}_p(z)$.

It remains to show injectivity of f. For this, it suffices (by (25)) to show for all $z \in \mathcal{D}_i(X), z' \in \mathcal{D}_{i'}(X)$ with $i, i' \in Min(\mathbb{I})$: If there is a proper mapping from z to z' in $\mathcal{D}$, then there is already a branching path from z to z'. We can assume $z \neq z'$, otherwise there is the empty branching path (cf. construction of reflexive closure in step 3 above).

Let thus

$$P = [(z = z_0, \delta_0, z_1), \ldots, (z_{n-1}, \delta_{n-1}, z_n = z')]$$

be such a proper path. Because $\mathbb{I}$ contains no directed cycles and z, z' are located in minimal components, we must have $n \geq 2$ and $\delta_0 \in \mathbb{I}_1^{op}$ and $\delta_{n-1} \in \mathbb{I}_1$. Because P is proper, it contains a positive number of branching positions, cf. Def.24. At each such position, the element z_j on the path gives rise to branching index ι_j.

The claim can now be proven by induction over the number t of branching positions: if $t = 1$, then P is already a branching path, namely a primary mapping path (step 2). Let $t + 1 \geq 2$ be the number of branching positions, j_1 the first (smallest) and j_2 the second position therein. Thus there is unique k with $j_1 < k < j_2$ and $\delta_{k-1} \in \mathbb{I}_1$, $\delta_k \in \mathbb{I}_1^{op}$.

There is an edge sequence q in $\mathbb{I}$ from vertex ι_k to a minimal index i''. Then the subpath of P from z_0 to z_k concatenated with the path Q from z_k to $\mathcal{D}_q(z_k)$ representing q is a primary branching path R (cf. step 2 above). And the reverse of Q concatenated with the subpath of P from z_k to z' can be made into a proper mapping path (by Lemma 29) with a number of branching positions $\leq t$, which connects $\mathcal{D}_q(z_k)$ and z'. Since these two elements are in minimal components $\mathcal{D}_{i''}$, $\mathcal{D}_{i'}$, resp., induction hypotheses yields a branching path from $\mathcal{D}_q(z_k)$ to z'. This path appended to R is a branching path from z to z'.[15] □

Note, that all the components $\mathcal{D}_i$ with $i \in Min(\mathbb{I})$ not being the target of any branch are not affected by the quotient construction, i.e. congruence classes $[z]_{\cong}$ are singletons for all $z \in \mathcal{D}_i(X)$. In Example 23 this is the case for index 3. Let $Af(\mathbb{I})$ denote the set of all affected minimal indices. We could then be even more specific and construct the colimit object as

$$\Big(\coprod_{i \in Min(\mathbb{I}) \setminus Af(\mathbb{I})} \mathcal{D}_i\Big) + \Big(\coprod_{i \in Af(\mathbb{I})} \mathcal{D}_i\Big) / \cong \tag{26}$$

6.4 Efficient Checking of the Van Kampen Property

Based on (26) we can conclude, independent of Corollary 20, that a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ is VK if $\mathbb{I}$, in addition of being finite and having no directed cycles, does not have branching indices either. In this case, $Af(\mathbb{I})$ is empty and the colimit of the diagram is simply given by the coproduct of all

[15] The so constructed path is - in general - not proper. But this is no problem here, because we did not claim this property for branching paths.

minimal components, thus the VK property of the diagram is ensured by the VK property of coproducts (extensivity) and pullback composition. In the presence of branching, however, we do not have VK for free. We have to check one of the conditions in Cor.20 for every single diagram.

The specialized colimit construction (26) suggests another practical relevant possibility to reduce our effort for checking VK in case $\mathbb{I}$ has no directed cycles. In applications that deal with nets of software components (e.g. multimodels), there is usually only one type of relation between the components: Relations either specify sameness of model elements, versions of one model element in evolving environments, or elements to be preserved when applying transformation rules [5]. Thus, rarely will it be the case that there are two or more morphisms in the same direction between two given components. An even weaker and also reasonable claim for two different relations is that they don't interfere in common codomains, thus the following definition is not too restrictive:

Definition 25 (Image-Disjointness) *A diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ is called* image disjoint*, if for each pair of different branches $p = (i \to \ldots \to j)$, $p' = (i \to \ldots \to j')$ in $\mathbb{I}$ starting in the same branching index i and all elements $y \in \mathcal{D}_i(X)$, $X \in \mathbb{B}$ we have $\mathcal{D}_p(y) \neq \mathcal{D}_{p'}(y)$.*

Clearly, by Cor.20, one obtains:

Fact 26 *If $\mathcal{D}$ is not image-disjoint, the colimt $\mathcal{D} \Rightarrow \Delta S$ does not have the Van Kampen property.*

because the two different branches p, p' and $y \in \mathcal{D}_i(X)$ with $\mathcal{D}_p(y) = \mathcal{D}_{p'}(y)$ yield two different mapping paths from y to $\mathcal{D}_p(y)$. □

If there are no undirected cycles in $\mathbb{I}$, then we have image disjointness for free, because we always assume that all components $\mathcal{D}_i$ are pairwise disjoint. If there are undirected cycles in $\mathbb{I}$, as in the case of coequalizers, for example, it can happen that $j = j'$. Thus we have to test, first, for image disjointness before the "different paths criterion for paths connecting affected minimal components" below can be applied. Note, that image disjointness implies that we have $\mathcal{D}_p(y) \neq \mathcal{D}_{p'}(y)$ for all $y \in \mathcal{D}_i(X)$, $X \in \mathbb{B}$ not only for branches but for arbitrary pairs of paths $p = (i \to \ldots \to j)$, $p' = (i \to \ldots \to j')$ starting in a common branching index $i \in Br(\mathbb{I})$ but not necessarily ending at a minimal index.

We can now outline a proof for the following theorem:

Theorem 27 (Different Paths Connecting Affected Minimal Components) *Let $\mathbb{G} = Set^{\mathbb{B}}$ and $\mathcal{D} : \mathbb{I} \to \mathbb{G}$ be a diagram with $\mathbb{I}$ a finite directed multigraph without directed cycles. Let*

$$\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$$

be a colimiting cocone with image-disjoint $\mathcal{D}$. The following are equivalent:

(1) *The cocone has the Van Kampen property*
(2) $\forall X \in \mathbb{B}, i, j \in Af(\mathbb{I}), z \in \mathcal{D}_i(X), z' \in \mathcal{D}_j(X)$: *There are no two different proper paths from z to z'*
(3) $\forall X \in \mathbb{B}, i, j \in Af(\mathbb{I}), z \in \mathcal{D}_i(X), z' \in \mathcal{D}_j(X)$: *There are no two different inner-cycle free proper paths from z to z'*

Thus – according to (24) – it is only necessary for an algorithm to iterate over branching components and compute paths into potentially affected minimal components. For instance, one has to consider only the small components $\mathcal{D}_{12}, \mathcal{D}_{13}, \mathcal{D}_{23}$ in Fig.1. Again the implementation is independent of whether it ignores inner-cycle free paths or not. It is, however, not guaranteed to find disjoint paths, if VK is violated.
Proof of Theorem 27: Obviously $(1) \Rightarrow (2)$ follows from Cor.20 and $(2) \Rightarrow (3)$ is trivial. Thus it remains to show $(3) \Rightarrow (1)$. For this, assume to the contrary that the Van Kampen property is not satisfied.

Thus, from (3) in Cor.20 there are two disjoint proper paths connecting some $z, z' \in \coprod_{i \in \mathbb{I}_0} \mathcal{D}_i$. Then the concatenation of the first and the reversed second path connects z with itself, yielding a new (cyclic) path

$$P = [(z = z_0, \delta_0, z_1), \ldots, (z_{n-1}, \delta_{n-1}, z_n = z)]$$

The remaining path is non-empty and still proper by disjointness of the given paths. It can be made inner-cycle free by Lemma 31. Since $\mathbb{I}$ has no loops, we have $n \geq 2$. Since $\mathbb{I}$ has no directed cycles, at least one of the δ_i's in P is in $\mathbb{I}_1$ and another one in $\mathbb{I}_1^{op}$. W.l.o.g. we can assume $\delta_0 \in \mathbb{I}_1^{op}$ and $\delta_{n-1} \in \mathbb{I}_1$. Because P is proper, there are smallest and largest branching positions $i \geq 1$ and $j \leq n-1$. We claim that $i \neq j$: If this would not be the case, then $(\delta_i = \delta_j) \neq (\delta_{i-1}^{op} = \delta_{j-1}^{op})$. Image disjointness yields $z_0 = \mathcal{D}_{\delta_0^{op}} \circ \ldots \circ \mathcal{D}_{\delta_{i-1}^{op}}(z_i) \neq \mathcal{D}_{\delta_{n-1}} \circ \ldots \circ \mathcal{D}_{\delta_j}(z_j) = z_n$, contradicting the assumption $z_0 = z_n$. Thus $i < j$, where $2 \leq j - i$ since $\delta_i \in \mathbb{I}_1$ and $\delta_{j-1} \in \mathbb{I}_1^{op}$.

We consider the unique $i < k < j$ with $\delta_x \in \mathbb{I}_1$ for all $i \leq x \leq k-1$ and $\delta_k \in \mathbb{I}_1^{op}$. There are edge sequences q_0 and q_k from ι_0 and ι_k to minimal indices i_0 and i_k, resp. Since ι_i is a branching index, by construction, we do have $i_0, i_k \in Af(\mathbb{I})$. Let Q_0, Q_k denote the mapping paths that represent the assignments $z_0 \mapsto \mathcal{D}_{q_0}(z_0)$ and $z_k \mapsto \mathcal{D}_{q_k}(z_k)$.

We can now construct two mapping paths connecting the element $\mathcal{D}_{q_k}(z_k)$ with the distinct (by image-disjointness at branching index ι_i) element $\mathcal{D}_{q_0}(z_0)$: The first mapping path R_1 is obtained by composing the reverse of Q_k with the reverse of the subpath, connecting z_0 with z_k, and then with Q_0. The second mapping path R_2 is obtained by composing the reverse of Q_k with the subpath connecting z_k with $z_n = z_0$, and then with Q_0.

It remains to show that R_1 and R_2 are inner-cycle free, since then we would have created a contradiction to premise (3). Obviously R_1 is inner-cycle free: Since P was inner-cycle free, it is only possible to have $y = y'$ at two different positions of R_1, if (w.l.o.g.) y is on Q_0. y' can not be on Q_0 and not on the subpath from z_0 to z_i, because this would yield a directed cycle in $\mathbb{I}$. But y' can also not reside on the subpath from z_i to $\mathcal{D}_{q_k}(z_k)$ because this would violate image-disjointness at branching position i.

R_2 can be reduced to an inner-cycle free path by Lemma 31. It is now important to verify that this reduction is not equal to R_1: A reduction of R_2 can only coincide with R_1, if their respective parts on P coincide. Because P was proper, this is only possible if the parts of P are totally erased during reduction. This, however, is not possible, because R_1 was not reduced. This, finally, yields two different proper and inner-cycle free paths as desired. □

The absence of directed cycles and the presence of image-disjointness are reasonable requirements for many practical use-cases. But circumstances can often be further narrowed. In the rest of this section we consider some other possibly satisfied properties and corresponding alternative checking methods which can simplify VK verification.

Monomorphisms: In some practical cases, the morphisms of $\mathcal{D}$ specify relations between components $\mathcal{D}_i$ and $\mathcal{D}_j$ such that an element in $\mathcal{D}_i$ is related to at most one element of $\mathcal{D}_j$. In this case all the morphisms $\mathcal{D}_d$, with d an edge in $\mathbb{I}$, are monomorphisms. In such a diagram any mapping path P is completely determined by y_0 (or y_n) and the corresponding sequence $[\delta_0, ..., \delta_{n-1}]$ of edges and opposed edges in $\mathbb{I}$. Due to Theorem 27, the diagram may be not VK only if there are two different sequences of edges and opposed edges between two distinct affected indices in $\mathbb{I}$. As long as there are no undirected cycles in $\mathbb{I}$ this can not happen, thus the diagram is VK, if there are no undirected cycles in $\mathbb{I}$.

If there are undirected cycles in $\mathbb{I}$ it is surely not enough to require image-disjointness as defined in Def.25, see also Fig.1. Instead, we can ensure VK by the stronger requirement that all the undirected

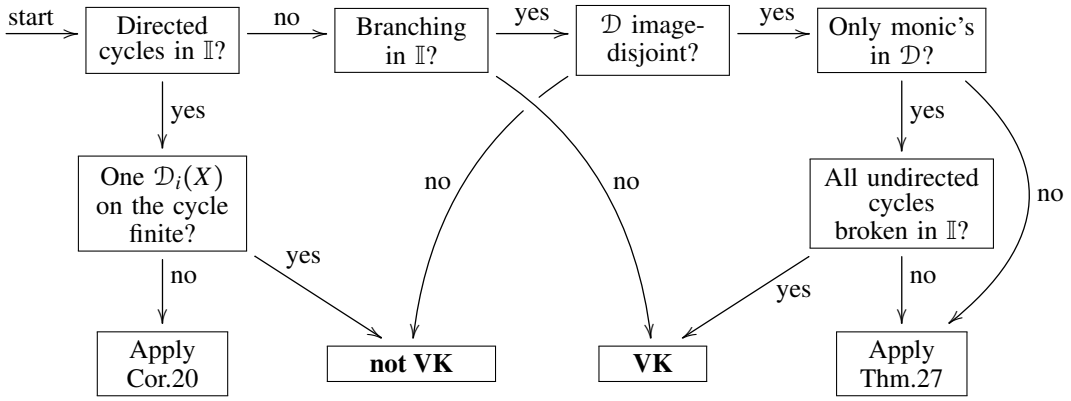


Figure 4: Decision diagram

cycles in $\mathbb{I}$ are broken in $\mathcal{D}$: An undirected cycle of edges in $\mathbb{I}$ is **broken in** $\mathcal{D}$ if for one of the situations

$$\ldots \xrightarrow{d_{n-1}} \cdots \xrightarrow{d_0} \xleftarrow{d'_0} \cdots \xleftarrow{d'_{m-1}} \ldots$$

in the edge sequence with $1 \leq n, m$ the morphisms $\mathcal{D}_{d_0} \circ \ldots \circ \mathcal{D}_{d_{n-1}}$ and $\mathcal{D}_{d'_0} \circ \ldots \circ \mathcal{D}_{d'_{m-1}}$ are image disjoint.

In the example in Figure 1 this condition is not satisfied. In the example "parametrized specification with import", however, it is quite natural that $\mathcal{D}_{ib}$ and $\mathcal{D}_r$ are image disjoint since the "imported component" $\mathcal{D}_I$ is not part of the "parameter component" $\mathcal{D}_p$.

6.5 Decision Diagram and VK Verification Algorithm

In practical cases, as outlined e.g. in Sect.3.3, colimit computation is obligatory. Verification of the Van Kampen property must follow, if we want to verify compositionality. It would thus be a nice side effect to have a possibility to check VK simultanously with colimit computation such that there is no increase in time complexity! We will now shortly discuss, that with the results gained so far, this is indeed possible.

For this, let's summarize the outcome of the previous sections as decision algorithm, see Fig.4.

With the exception of rare cases in which there is a directed cycle in $\mathbb{I}$ whose component carrier sets are all infinite, it is possible to easily reach an early decision, if either there are directed cycles in $\mathbb{I}$ or if there is no branching in $\mathbb{I}$. These analysis is restricted to the small graph $\mathbb{I}$. In the presence of branching, the natural next step is to check violation of image-disjointness in $\mathcal{D}$ to immediately deduce violation of VK (Fact 26). Image-disjointness can immediately be confirmed, if all branches diverge. Otherwise, the mapping behavior along branches has to be investigated, which may be more costly.

Hence the *combined algorithm for colimit computation and Van Kampen verification* comprises the following steps:

1. Preprocessing of the data shows whether we are on a decision route in Fig.4 on which Thm.27 will be applied. In this case there are *no directed cycles in* $\mathbb{I}$ and $\mathcal{D}$ is *image-disjoint*.
2. $vk := true$;
3. $\cong_X := \{(z,z) \mid z \in \mathcal{D}_j(X),\ j \in Min(\mathbb{I})\}$ for all $X \in \mathbb{B}$;
4. For each branching component $\mathcal{D}_i$, each $X \in \mathbb{B}$ and for each $y \in \mathcal{D}_i(X)$, do:

(a) Add images (z,z') to $\cong_X$ according to primary identifications in step 2 in the specialized colimit computation.
(b) Keep $\cong_X$ transitive by adding all arising transitive pairs from the last enhancement.
(c) Whenever in the two previous steps a pair (z,z') is added for the second time, $vk := false$ (cf. Theorem 27).

5. Compute colimit cocone κ as in (24) using the family $\cong = (\cong_X)_{X \in \mathbb{B}}$.
6. Return (κ, vk)

While, in such a way, VK verification can be embedded into the obligatory colimit computation, the main disadvantage is the possibly costly preprocessing, especially the early image-disjointness check. In the next section, we describe an alternative and slightly deviating approach, how to avoid this check. The drawback, however, will be, that this method can not directly be embedded into the colimit computation.

6.6 Checking for Cyclic Mapping Paths only

All conditions in Corollary 20 are equivalent to

$$\forall z: \text{ There is no non-empty proper mapping path connecting } z \text{ with itself,} \tag{27}$$

because on the one hand any cyclic proper non-empty mapping path from z to itself is accompanied by the empty path from z to itself. On the other hand, if $z \neq z'$, the existence of two disjoint proper mapping paths in $\mathcal{D}$ connecting z and z' provides a non-empty proper mapping path connecting z with itself since the first path can be composed with the reversed second path. This composition becomes proper again since the paths are disjoint.

If $\mathbb{I}$ has no directed cycles, we will show now that it is even enough to check only the branching components for cyclic mapping paths. Note, that this criterion generalizes smoothly our original condition in Theorem 10 for VK pushouts. In the same way as Theorem 27, this criterion is also useful in practice, because, again, one can reduce analysis to presumably small branching components.

We consider a non-empty mapping path $P = [(y_0, \delta_0, y_1), (y_1, \delta_1, y_2), \ldots, (y_{n-1}, \delta_{n-1}, y_n)]$ with $y_0 = y_n$. If $\iota_0 \notin Br(\mathbb{I})$ we can not have $\delta_{n-1} \in \mathbb{I}_1^{op}$ and $\delta_0 \in \mathbb{I}_1$ at the same time. Since there is no loop in $\mathbb{I}$ we have $n \geq 2$. Moreover, since there are no directed cycles in $\mathbb{I}$ at least one of the δ_i's must be in $\mathbb{I}_1$ and at least one in $\mathbb{I}_1^{op}$. In such a way, there must be at least one branching position j with $0 < j < n$, i.e. ι_j is a branching index. Thus the cycle can be transformed into a non-empty cyclic proper mapping path connecting y_j in $\mathcal{D}_{\iota_j}(X)$ with itself where $\iota_j \in Br(\mathbb{I})$. This shows that we need to check condition (27) indeed only for branching indices. Note, that we have, in such a way, another argument, now based on Corollary 20, that we have VK for free if there is no branching in our finite $\mathbb{I}$ without directed cycles!

Due to inner-cycle reduction (Lemma 31) it would be even enough to check the existence of cyclic non-empty inner-cycle free and proper mapping paths for elements in branching components. This yields

Theorem 28 (Non-Empty Cyclic Paths) *Let $\mathbb{G} = Set^{\mathbb{B}}$ be a presheaf topos and $\mathcal{D}: \mathbb{I} \to \mathbb{G}$ be a diagram with $\mathbb{I}$ a finite directed multigraph without directed cycles. Let*

$$\mathcal{D} \overset{\kappa}{\Rightarrow} \Delta S$$

be a colimiting cocone. The following are equivalent:

(1) *The cocone is VK*
(2) $\forall X \in \mathbb{B}, i \in Br(\mathbb{I}), z \in \mathcal{D}_i(X)$: *There is no non-empty proper mapping path from z to itself*
(2) $\forall X \in \mathbb{B}, i \in Br(\mathbb{I}), z \in \mathcal{D}_i(X)$: *There is no non-empty inner-cycle free and proper mapping path from z to itself* □

In Fig.4 it is now possible to omit the check for image-disjointness and carry out the algorithm according to Theorem 28 after having checked for directed cyles and branching in $\mathbb{I}$. In Fig.1, it is easy to see that the cyclic path

$$(Sort, d^{op}_{-13}, S/I), (S/I, d_{13}, Intf.), (Intf., d^{op}_{23}, T/I), (T/I, d_{-23}, Type), (Type, d^{op}_{12}, S/T), (S/T, d_{-12}, Sort)$$

is the witness for VK violation. Not surprisingly, it is the concatenation of the two indicating paths already given in Sect.3.3.

The methodology of the present section, however, is not directly combinable with colimit computation, since the algorithm of Sect.6.5 collects pairs in minimal components only, while cyclic path construction takes place in arbitrary components.

7 Conclusion

Our main interest as theoreticians is to find nice general results. Thus we have a tendency to abstract away problematic features and aspects and to concentrate on simplified structures and problems. In addition, our focus may be rather guided by inner-theoretic interests than by practical problems.

Practitioners are, in contrast, often confronted with complex problems and structures. Moreover, the practical problems have to be solved even if no general solutions are available. Even if (the theoreticians say that) there is no general solution there may be still a satisfactory solution for the practical problem in question.

In general, arbitrary diagrams in arbitrary categories are not VK. Even if we restrict to presheaf topoi, most of the diagrams are not VK. In the paper we presented and proved a feasible general necessary and sufficient condition to check if a diagram in a presheaf topos is VK or not. Based on this general result, we developed general practical criteria to check VK for a spectrum of different special kinds of diagrams.

The definition of *adhesiveness* guarantees VK for pushouts, if one morphism in the span is monic, but there seems to be no natural and general way to generalize adhesiveness to arbitrary diagrams. A first attempt to define such a concept for a diagram $\mathcal{D} : \mathbb{I} \to Set^{\mathbb{B}}$ may be that for any branching index $i \in Br(\mathbb{I})$ there is, at most, one branch $p = (i \to \cdots \to j)$ such that $\mathcal{D}_p$ is not monic. This is, however, by far not enough: For the diagram $\mathcal{D}_2 \overset{\mathcal{D}_{d_1}}{\leftarrow} \mathcal{D}_0 \overset{\mathcal{D}_{d_2}}{\to} \mathcal{D}_3 \overset{\mathcal{D}_{d_3}}{\leftarrow} \mathcal{D}_1 \overset{\mathcal{D}_{d_4}}{\to} \mathcal{D}_4$ we do have VK if ($\mathcal{D}_{d_1}$ and $\mathcal{D}_{d_3}$) or ($\mathcal{D}_{d_2}$ and $\mathcal{D}_{d_4}$) are monic. In case, $\mathcal{D}_{d_2}$, $\mathcal{D}_{d_3}$ monic and $\mathcal{D}_{d_1}$, $\mathcal{D}_{d_4}$ not monic, however, we may not have VK. In this case, we can force VK by requiring that $\mathcal{D}_{d_2}$ and $\mathcal{D}_{d_3}$ are image disjoint. Otherwise, we have to check one of our criteria. In other examples, e.g. "parametrized specification with import", we will still have VK if we allow $\mathcal{D}_{is}$ and (either $\mathcal{D}_m$ or $\mathcal{D}_r$) to be not monic as long as $\mathcal{D}_{ib}$ and $\mathcal{D}_r$ are image disjoint. As the above example shows, even the absence of undirected cycles is not of much help to achieve VK. It is necessary to find appropriate "breaking requirements" depending on the class of graph $\mathbb{I}$.

In practice, however, it may be even not necessary that a given diagram (of software models) is VK, i.e., that indeed all instances of the diagram have to be amalgamable. Or, practitioners may insist on things like "multiple inheritance" or redundancy, as in the example in Figure 1 and thus take into account to loose overall amalgamability. They may say that they are clever enough to avoid the "twisting anomalies", the theoreticians are concerned about, and to ensure that all the appearing instances of the diagram of models are amalgamable. It is maybe worth to underline that all the instances of a diagram that we get by decomposing instances of the compound colimit model, are amalgamable. Also the instances we get from a given "indexed semantics" via a corresponding variant of the Grothendieck construction [29] are amalgamable.

The natural next step, to meet more practical challenges, will be to look for feasible conditions that a given instance of a diagram is amalgamable (even if the diagram is not VK). We want to tell the practitioners in what sense they have to be careful and clever. The essential heuristics is that "semantics should be handled and controlled by syntax". We may allow cyclic paths in diagrams of models but then any cyclic path in the diagram of instances has to be the exact copy of a cyclic path in the diagram of models.

Following these heuristics, we have presented in [30] a general condition for amalgamability of instances of pushout diagrams in presheafs. There seem to be no principal problems to transform this condition into an amalgamability condition for instances of arbitrary colimits in presheaf topoi along the argumentations developed in the present paper. It may only be technically involved and time consuming.

The ultimate goal, however, is to find a categorical counterpart for the different paths criterion Cor.20, 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 formulated in [12]?

8 Appendix: Reduction Techniques

Lemma 29 (Path Reduction) *Given a diagram $\mathcal{D}:\mathbb{I}\to\mathbb{G}$ in $\mathbb{G}=Set^{\mathbb{B}}$, any mapping path P of sort $X\in\mathbb{B}$ connecting y_0 and y_n can be reduced to a proper mapping path connecting y_0 and y_n.*

Proof: Since a path may contain different cycles we have to prove by induction over the length of paths.
Basic cases: All paths of length 0 or 1 are proper.
Induction Step: Assume the path is not proper, i.e., we have especially $n\geq 2$. We first choose the minimal $0\leq i<n-1$ such that there is an $i<j\leq n-1$ with $(y_i,\delta_i,y_{i+1})=_w(y_j,\delta_j,y_{j+1})$. And second, we choose the maximal of those j's for the chosen minimal i.
Case 1: $(y_i,\delta_i,y_{i+1})=(y_j,\delta_j,y_{j+1})$: The reduced mapping path is

$$[(y_0,\delta_0,y_1),...,(y_i,\delta_i,y_{i+1}),(y_{j+1},\delta_{j+1},y_{j+2}),\ldots,(y_{n-1},\delta_{n-1},y_n)].$$

It still connects y_0 with y_n. This also holds in the exceptional case $j=n-1$, because then the last segment in the sequence is (y_i,δ_i,y_{i+1}) in which $y_{i+1}=y_{j+1}=y_n$.
Case 2: $(y_i,\delta_i,y_{i+1})=(y_{j+1},\delta_j^{op},y_j)$. The path

$$[(y_0,\delta_0,y_1),...,(y_{i-1},\delta_{i-1},y_i),(y_{j+1},\delta_{j+1},y_{j+2}),\ldots,(y_{n-1},\delta_{n-1},y_n)] \tag{28}$$

connects y_0 with y_n. This also holds in the exceptional cases $i=0$ (then the first part is empty and $y_{j+1}=y_i=y_0$) and $j=n-1$ (then in the path in (28) the second part is already empty and $y_n=y_{j+1}=y_i$ [16]). In both cases cycles occur in the part starting at index $j+1$ only (by the choice of i). Since the length of this part is $n-j-1<n$, application of induction hypotheses (reduction of the second part) yields a proper path. □

Lemma 30 (Enforcing Disjointness) *Given a diagram $\mathcal{D}:\mathbb{I}\to\mathbb{G}$ in $\mathbb{G}=Set^{\mathbb{B}}$ and $X\in\mathbb{B}$. Let $z_1,z_1'\in\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i(X)$ and P and Q be two different proper mapping paths of sort X in $\mathcal{D}$ connecting z_1 and z_1', resp. Then there are $z_2,z_2'\in\coprod_{i\in\mathbb{I}_0}\mathcal{D}_i(X)$ and two disjoint proper paths both connecting z_2 and z_2'.*

[16] *$i=0$ and $j=n-1$* means that both parts and hence the whole path in the reduction is empty. Then y_0 is also connected with y_n, because $y_n=y_{j+1}=y_i=y_0$, cf. Def.5.

Proof:: We can assume that P_1 and P_2 are non-empty. Otherwise they are disjoint and the statement holds with $z_2 = z_2' = z_1$. Let $P = [S_0^P, \ldots, S_{n-1}^P]$ and $Q = [S_0^Q, \ldots, S_{m-1}^Q]$. We can assume that S_0^P and S_0^Q are already different (not equal according to Def.5). Otherwise we shorten both paths by cutting off leading identical segments up to a segment starting at some z_2.

Let then $i \geq 0$ be the smallest index such that there is (a smallest) $j \geq 1$ with $S_i^P =_w S_j^Q$. Thus the paths

$$P' = [S_0^P, \ldots, S_i^P] \text{ and } Q' = [S_0^Q, \ldots, S_{j-1}^Q]$$

are disjoint and proper. Let $S_i^P = (y, \delta, y')$.

Case 1: $S_i^P = S_j^Q$. Then S_{j-1}^Q has the form $(\_, \_, y)$. If $i > 0$ then S_{i-1}^P is also of this form, thus $[S_0^P, \ldots, S_{i-1}^P]$ and $[S_0^Q, \ldots, S_{j-1}^Q]$ are disjoint and connect z_2 with $z_2' := y$. If $i = 0$, then P and hence also Q start at y, thus $[S_0^Q, \ldots, S_{j-1}^Q]$ is cyclic, hence disjoint from $[\,]$ (i.e. we take $z_2' = z_2 = y$).

Case 2: $S_i^P \neq S_j^Q$, i.e. $S_j^Q = (y', \delta^{op}, y)$, then S_{j-1}^Q is $(\_, \_, y')$ and we take P' and Q' (i.e. $z_2' = y'$). □

The following lemma was needed in order to get rid of properness of mappping paths in the main theorem.

Lemma 31 (Inner-Cycle Deletion) *Any non-empty proper mapping path, connecting y_0 and y_n, can be reduced to a non-empty inner-cycle free proper mapping path connecting y_0 and y_n.*

Proof: We use the same technique as in the proof of Lemma 30: If there is an inner cycle we choose the smallest index $0 \leq i$ such that there is an $i < j \leq n$ with $j - i \leq n - 1$ and $y_i = y_j$. Second, we choose the maximal of those j's for the chosen minimal i and reduce accordingly. □

Lemma 32 (Domain Cycle Reduction) *Given a span $\mathcal{D}_1 \xleftarrow{h_1} \mathcal{D}_0 \xrightarrow{h_2} \mathcal{D}_2$ in $\mathbb{G} = Set^{\mathbb{B}}$ any domain cycle $[x_0, x_1, \ldots, x_{2k+1}]$ of sort $X \in \mathbb{B}$ can be reduced to a proper domain cycle.*

Proof: Since a domain cycle may contain different smaller domain cycles we have to prove by induction over the length of domain cycles.
Basic cases: Due to condition (1) in Definition 9 all domain cycles with $k = 0$, i.e., of length 2, are proper.
Induction Step: Assume that the domain cycle is not proper, i.e., we have especially $k \geq 1$. We choose first the minimal $0 \leq i < 2k+1$ such that there is an $i < j \leq 2k+1$ with $x_i = x_j$. And second, we choose the maximal of those j's for the chosen minimal i. Due to condition (1) in Definition 9 we can not have $j - i = 1$ and we can not have $i = 0$ and $j = 2k+1$ since $x_{2k+1} \neq x_{2k+2} = x_0$. In such a way, we get for the difference $2 \leq (j-i) \leq 2k$.
Case 1 $(j-i) = 2m$ with $1 \leq m \leq k$: In this case we have $x_i = x_j \neq x_{j+1}$ and it can be easily checked that the sequence $[x_0, \ldots, x_i, x_{j+1}, \ldots, x_{2k+1}]$ also satisfies conditions (2) and (3) in Definition 9. In such a way, we obtain a reduced domain cycle of length $(2k+2) > (2k+2) - (i-j) = (2k+2) - 2m = 2(k-m) + 2 \geq 2$ that is proper due to the choice of i and j.
Case 2 $(j-i) = 2m+1$ with $1 \leq m < k$: If $0 < i$ we have $x_{i-1} \neq x_{j+1}$ due to the choice of i and it can be easily checked that the sequence $[x_0, \ldots, x_{i-1}, x_{j+1}, \ldots, x_{2k+1}]$ also satisfies conditions (2) and (3) in Definition 9. In such a way, we obtain a reduced domain cycle of length $(2k+2) > (2k+2) - ((i-j)+1) = (2k+2) - (2m+2) = 2(k-m) \geq 2$ that is proper due to the choice of i and j.
If $i = 0$ we obtain the reduced domain cycle $[x_{j+1}, \ldots, x_{2k+1}]$ of length $(2k+2) > (2k+1) - j = (2k+1) - (2m+1) = 2(k-m) \geq 2$. This reduced domain cycle may be not proper and we have to apply the induction hypothesis. Note, that x_0 is no longer in the reduced domain cycle. □

We draw attention to the fact that in Case 2, $i = 0$ the reduction of the corresponding weak mapping path, according to Lemma 30 does not produce a domain cycle.

References

[1] M. Barr & C. Wells (1990): *Category Theory for Computing Sciences*. Prentice Hall.

[2] R. Brown & G. Janelidze (1997): *Van Kampen Theorems for Categories of Covering Morphisms in Lextensive Categories*. *J. Pure Appl. Alegbra* 119, pp. 255 – 263.

[3] M. Bunge & S. Lack (2003): *van Kampen Theorems for Topoi*. *Advances in Mathematics* 179, pp. 291 – 317.

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

[5] H. Ehrig, K. Ehrig, U. Prange & G. Taentzer (2006): *Fundamentals of Algebraic Graph Transformations*. Springer.

[6] Hartmut Ehrig, M. Grosse-Rhode & U. Wolter (1998): *Applications of Category Theory to the Area of Algebraic Specification in Computer Science*. *Appl. Categorical Structures* 6, pp. 1–35.

[7] Jose Luiz Fiadeiro (2005): *Categories for Software Engineering*. Springer.

[8] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein & M. Goedicke (1992): *Viewpoints: A Framework for Integrating Multiple Perspectives in System Development*. *International Journal of Software Engineering and Knowledge Engineering* 2.

[9] M. Fowler & K. Scott (1999): *UML distilled*. Addison-Wesley.

[10] Peter Freyd (1972): *Aspects of Topoi*. *Bull. Austral. Math. Soc.* 7, pp. 1–76, doi:10.1017/S0004972700044828.

[11] Robert Goldblatt (1984): *Topoi: The Categorial Analysis of Logic*. Dover Publications.

[12] T. Heindel & P. Sobocinski (2009): *Van Kampen Colimits as Bicolimits in Span*. In A. Kurz, M. Lenisa & A. Tarlecki, editors: *Algebra and Coalgebra in Computer Science*, *Lecture Notes in Comput. Sci.* 5728, Springer Berlin / Heidelberg, pp. 335–349, doi:10.1007/978-3-642-03741-2_23.

[13] Wolfram Kahl (2011): *Collagories: Relation-algebraic reasoning for gluing constructions*. *J. Log. Algebr. Program.* 80(6), pp. 297–338, doi:10.1016/j.jlap.2011.04.006. Available at `http://dx.doi.org/10.1016/j.jlap.2011.04.006`.

[14] E. R. van Kampen (1933): *On the Connection between the Fundamental Groups of some Related Spaces*. *American Journal of Mathematics* 55, pp. 261 – 267.

[15] Harald König, Michael Löwe, Christoph Schulz & Uwe Wolter (2014): *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*, pp. 222–236, doi:10.1007/978-3-319-09108-2_15. Available at `http://dx.doi.org/10.1007/978-3-319-09108-2_15`.

[16] S. Lack & P. Sobociński (2004): *Adhesive Categories*. In: *Foundations of Software Science and Computation Structures (FoSSaCS '04)*, 2987, Springer, pp. 273–288, doi:10.1007/978-3-540-24727-2_20.

[17] S. Lack & P. Sobociński (2006): *Toposes are Adhesive*. *Lecture Notes in Comput. Sci.* 4178, pp. 184–198, doi:10.1007/11841883_14.

[18] Michael Löwe (2010): *Van-Kampen Pushouts for Sets and Graphs*. Technical Report, University of Applied Sciences, FHDW Hannover.

[19] Moerdijk I. Mac Lane, S. (1992): *Sheaves in Geometry and Logic A first introduction to topos theory*. Springer.

[20] Saunders Mac Lane (1998): *Categories for the Working Mathematician, Second edition*. Springer.

[21] M. Makkai (1997): *Generalized sketches as a framework for completeness theorems*. *J. Pure Appl. Algebra* 115, pp. 49–79, 179–212, 214–274.

[22] J.P. May (1999): *A Concise Course in Algebraic Topology*. Chicago Lectures in Mathematics, The University of Chicago Press. Available at `http://dx.doi.org/10.1007/978-3-642-17336-3`.

[23] M. Sabetzadeh, S. Nejati, S. Liaskos, S. Easterbrook & M. Chechik (2007): *Consistency Checking of Conceptual Models via Model Merging*. In: *RE*, pp. 221–230.

[24] Mehrdad Sabetzadeh, Shiva Nejati, Sotirios Liaskos, Steve M. Easterbrook & Marsha Chechik (2007): *Consistency Checking of Conceptual Models via Model Merging*. In: *RE*, IEEE, pp. 221–230. Available at `http://dx.doi.org/10.1109/RE.2007.18`.

[25] Donald Sannella & Andrzej Tarlecki (2012): *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series, Springer, doi:10.1007/978-3-642-17336-3. Available at `http://dx.doi.org/10.1007/978-3-642-17336-3`.

[26] Herbert Seifert (1931): *Konstruktion dreidimensionaler geschlossener Räume*. *Dissertation, University of Dresden* .

[27] P. Sobocziński (2004): *Deriving Process Congruences from Reaction Rules*. Technical Report DS-04-6, BRICS Dissertation Series.

[28] Angelo Vistoli (2005): *Notes on Grothendieck topologies, fibered categories and descent theory*. *arXiv:math/0412512V2* .

[29] U. Wolter & Z. Diskin (2007): *From Indexed to Fibred Semantics – The Generalized Sketch File –*. Reports in Informatics 361, Dep. of Informatics, University of Bergen.

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

UNIVERSITY OF APPLIED SCIENCES

FHDW

FACHHOCHSCHULE FÜR DIE WIRTSCHAFT
HANNOVER

ISSN 1863-7043