Article

Not peer-reviewed version

Applied Category Theory in the Wolfram Language using Categorica I: Diagrams, Functors and Fibrations

Jonathan Gorard *

Posted Date: 25 March 2024

doi: 10.20944/preprints202403.1479.v1

Keywords: Applied category theory; monoidal categories; Mathematica

Article

Applied Category Theory in the Wolfram Language Using CATEGORICA I: Diagrams, Functors and Fibrations

Jonathan Gorard

Princeton University, Princeton, NJ, USA; gorard@princeton.edu

Abstract: This article serves as a preliminary introduction to the design of a new, open-source applied and computational category theory framework, named CATEGORICA, built on top of the Wolfram Language. CATEGORICA allows one to configure and manipulate abstract quivers, categories, groupoids, diagrams, functors and natural transformations, and to perform a vast array of automated abstract algebraic computations using (arbitrary combinations of) the above structures; to manipulate and abstractly reason about arbitrary universal properties, including products, coproducts, pullbacks, pushouts, limits and colimits; and to manipulate, visualize and compute with strict (symmetric) monoidal categories, including full support for automated string diagram rewriting and diagrammatic theorem-proving. In so doing, CATEGORICA combines the capabilities of an abstract computer algebra framework (thus allowing one to compute directly with epimorphisms, monomorphisms, retractions, sections, spans, cospans, fibrations, etc.) with those of a powerful automated theorem-proving system (thus allowing one to convert universal properties and other abstract constructions into (higher-order) equational logic statements that can be reasoned about and proved using standard automated theorem-proving methods, as well as to prove category-theoretic statements directly using purely diagrammatic methods). Internally, CATEGORICA relies upon state-of-the-art graph and hypergraph rewriting algorithms for its automated reasoning capabilities, and is able to convert seamlessly between diagrammatic/combinatorial reasoning on labeled graph representations and symbolic/abstract reasoning on underlying algebraic representations of all category-theoretic concepts. In this first of two articles introducing the design of the framework, we shall focus principally upon its handling of quivers, categories, diagrams, groupoids, functors and natural transformations, including demonstrations of both its algebraic manipulation and theorem-proving capabilities in each case.

Keywords: applied category theory; monoidal categories; mathematica

1. Introduction

As a field of pure mathematics, category theory emerged in the mid-20th century with the work of Eilenberg and Mac Lane on group theory and algebraic topology [1,2], as well as the work of Serre and Grothendieck in the closely related area of homological algebra [3,4]. When conceived as an alternative foundation for mathematics (as exemplified by work on *elementary topos theory* within mathematical logic [5,6]), the shift from set theory to category theory necessitates a certain shift in philosophical perspective, wherein one transitions away from identifying mathematical objects based upon their internal structure (such as in Zermelo-Fraenkel set theory, where the axiom of extensionality defines set equality based purely upon the set membership relation [7]), and towards identifying mathematical objects based upon their relationships to other objects of the same "type" (as exemplified by the Yoneda lemma, arguably the most fundamental result in category theory [8]). This more *relational* perspective on the nature of structure turns out to be a useful lens through which to view many fields, including many outside of mathematics altogether. For instance, in the foundations of quantum mechanics, moving from set-theoretic to category-theoretic foundations inevitably changes one's view from quantum systems and their states as being the fundamental objects of study (as in the traditional Dirac-von Neumann axioms of quantum mechanics, represented in terms of operator theory on Hilbert space [9,10]) to quantum processes and their compositions as being the fundamental objects of study (as in the categorical quantum mechanics formalism of Abramsky and Coecke [11,12],

 doi:10.20944/preprints202403.1479.v1

later developed into a fully diagrammatic theory of quantum information by Coecke and Duncan [13,14]). Likewise, in computer science, one is able to move from a set-theoretic view in which program state is treated as fundamental (as in the Turing machine picture of computation, and as exemplified by the imperative programming paradigm) to a category-theoretic view in which functions and their compositions are treated as fundamental (as in the λ-calculus picture of computation, and as exemplified by the functional programming paradigm). Such process-theoretic models, in which processes and their algebra of composition are treated on a fundamentally "higher" footing than states and their internal structure, have also proved useful and/or instructive for the foundations of natural language processing [15], cybernetics [16], machine learning [17], complex networks and control theory [18,19], computational complexity theory [20] (based on prior work on computational irreducibility [21]), database systems [22], and many other domains of knowledge. This general process of applying foundational methods and concepts from category theory to fields outside of pure mathematics has come to be known as *applied category theory*, [23].

With the advent and maturation of so many fruitful domains of applied category theory, many software tools have now been developed to facilitate working with category-theoretic data structures in an automated or semi-automated fashion, such as CATLAB.JL [24], built on top of the Julia programming language, which uses the formalism of generalized algebraic theories and dependent types to facilitate automated algebraic manipulation of certain fundamental data structures such as operads and symmetric monoidal categories. There are also many examples of diagrammatic proof assistants, such as the highly general QUANTOMATIC [25] and CARTOGRAPHER [26] frameworks, or the more specialized PYZX framework [27], which facilitate the automated rewriting of string diagrams in symmetric monoidal category theory (with, in the latter case, a particular emphasis on diagrammatic quantum information theory). There are even projects, such as the ANR COREACT project, which aim to formalize certain key aspects of applied category theory, such as compositional/categorical rewriting theory, within existing proof assistant frameworks such as COQ. The aim of the present article is to introduce another such software framework, known as CATEGORICA, which is written principally in the Wolfram Language and is designed to be fully integrated into the Mathematica software system, and which seeks to combine many of the computational algebraic capabilities of existing frameworks such as CATLAB with many of the diagrammatic theorem-proving capabilities of frameworks such as QUANTOMATIC. At its core, the CATEGORICA framework consists of a unified collection of advanced symbolic and diagrammatic algorithms for efficiently converting between purely algebraic representations of categories, diagrams, functors and other key category-theoretic constructions (by employing an entirely presentation-theoretic view of categories, with arrows within an underlying quiver acting as generators, composition acting as the fundamental binary operation, and algebraic equivalences between objects and morphisms acting as relations) and purely graph-theoretic/combinatorial representations of the very same structures (by employing a description of quivers, categories and functors in terms of graphs, hypergraphs and combinatorial rewriting systems). In this manner, CATEGORICA is able to combine the features of a diagrammatic theorem-prover, a higher-order (equational) logic theorem prover, a (hyper)graph rewriting framework, and an abstract computer algebra system, all in an entirely seamless fashion, automatically adopting the most appropriate algorithmic approach and most efficient concrete data structures for any particular problem, and then converting the result back into the desired abstract representation at the end.

The first in this series of two articles introducing the design of CATEGORICA will focus primarily on the core algebraic structures of the framework (with a particular emphasis on CATEGORICA's representation and handling of quivers, categories, functors and natural transformations), some simple diagrammatic theorem-proving capabilities (such as CATEGORICA's ability to derive and prove necessary and sufficient algebraic conditions for diagrams to commute, for categories to be groupoids, for functors to be faithful, etc.), as well as some more advanced abstract algebraic capabilities (such as the computation of retractions and sections, epimorphisms and monomorphisms, initial and terminal objects, constant and coconstant morphisms, injectivity/surjectivity of functors on both

 doi:10.20944/preprints202403.1479.v1

objects and morphisms, and even some simple cases of Grothendieck fibrations). The forthcoming second article in the series will focus upon the extension of these capabilities to the handling of universal properties (especially products and coproducts, pullbacks and pushouts, and limits and colimits), including illustrations both of how to prove theorems using them, and of how to perform algebraic computations involving them, as well as the applications of these methods to the handling of (strict, symmetric) monoidal categories, including capabilities for string diagram representation, manipulation and rewriting. CATEGORICA's core algorithmic framework is based on the hypergraph rewriting/Wolfram model formalism outlined within [28–30], which may be augmented with basic techniques from the theory of graph grammars, and algebraic/compositional graph rewriting theory [31,32] (and, in particular, the theory of *double-pushout* rewriting [33]) in order to develop a highly efficient (hyper)graphical/diagrammatic theorem-proving system [34–36]. We begin in Section 2 by introducing the fundamental data structure underlying all abstract category-theoretic objects in CATEGORICA, namely the abstract *quiver* (a directed multigraph whose vertices are *objects* and whose edges are *arrows*), and we show how an `AbstractQuiver` object in CATEGORICA may be used in order to "freely generate" a corresponding `AbstractCategory` object by allowing for certain arrows to be composed (at which point they become *morphisms*), as well as introducing identity morphisms for each object, in a manner that is consistent with both associativity and identity axioms. We demonstrate how CATEGORICA is able to keep track of all necessary algebraic equivalences between morphisms (as necessitated by the underlying axioms of category theory) automatically, as well as how new algebraic relations between objects and arrows/morphisms may be introduced, in order to construct more general examples *non-free* categories. We also highlight a very simple application of CATEGORICA's automated theorem-proving capabilities, by proving that certain categorical diagrams *commute* (i.e. that, within certain categories, all directed paths yield the same morphism up to algebraic equivalence), and by automatically computing both necessary and sufficient algebraic conditions to force non-commuting categorical diagrams to commute, using the `AbstractCategory` function.

In Section 3, we proceed to show some examples of simple algebraic computations that can be performed using the `AbstractCategory` function, such as the computation of monomorphisms, epimorphisms and bimorphisms (i.e. the category-theoretic generalizations of injective, surjective and bijective functions in mathematical analysis), and the computation of sections, retractions and isomorphisms (i.e. the category-theoretic generalizations of left-invertible, right-invertible and invertible elements in abstract algebra). We introduce a special kind of category known as a *groupoid*, in which all morphisms are isomorphisms, and we show how CATEGORICA's automated theorem-proving capabilities can again be used in order to prove that certain `AbstractCategory` objects are or are not groupoids, and indeed to compute necessary and sufficient algebraic conditions (whenever they exist) to force non-groupoidal `AbstractCategory` objects to become groupoidal. The relationships between monomorphisms vs. epimorphisms and retractions vs. sections are both examples of a far more general category-theoretic notion of *duality*: the *dual* of a particular construction can be obtained by simply reversing the direction of morphisms, and therefore reversing the order in which morphisms are composed. CATEGORICA has in-built functionality (via the *"DualCategory"* property in particular) for making the systematic exploration of category-theoretic dualities completely straightforward, and we illustrate this by investigating some other common examples of dual constructions using CATEGORICA, including initial vs. terminal objects (i.e. the category-theoretic generalizations of bottom and top elements in partially ordered sets), strict initial vs. strict terminal objects (i.e. initial and terminal objects whose incoming/outgoing morphisms are all isomorphisms), and constant vs. coconstant morphisms (i.e. the category-theoretic generalizations of constant functions and zero-maps in mathematical analysis). In Section 4, we move on to explore functors (i.e. homomorphisms/structure-preserving maps between categories) using the `AbstractFunctor` function, demonstrating CATEGORICA's capabilities for handling both covariant and contravariant functors (i.e. functors in which morphism directions are preserved and reversed, respectively). We illustrate how CATEGORICA is able to distinguish between different notions of injectivity, surjectivity and bijectivity for `AbstractFunctor` objects, including

 doi:10.20944/preprints202403.1479.v1

injectivity, surjectivity and bijectivity on objects; essential injectivity, essential surjectivity and essential bijectivity (i.e. injectivity, surjectivity and bijectivity on objects, but only up to isomorphism); and faithfulness, fullness and full faithfulness (i.e. injectivity, surjectivity and bijectivity on *morphisms*). We also show CATEGORICA may be used to investigate an important class of functors known as Grothendieck fibrations (i.e. the category-theoretic generalizations of fiber bundles in topology) from a total category to a base category, including special cases such as discrete fibrations (i.e. Grothendieck fibrations in which the fiber categories are all discrete categories containing only objects and identity morphisms).

Finally, in Section 5, we introduce the `AbstractNaturalTransformation` function, allowing one to represent arbitrary natural transformations between `AbstractFunctor` objects in CATEGORICA, and showcase its ability to compute (and prove) both necessary and sufficient algebraic conditions to force transformations between functors to be natural. We demonstrate furthermore how the `AbstractNaturalTransformation` framework may be used to detect natural isomorphisms between both objects and functors in a highly general way. We also include a short note regarding some of the internal algorithmic details of the CATEGORICA system, and, in particular, its use of double-pushout rewriting formalism and compositional graph rewriting theory to reduce abstract algebraic and diagrammatic reasoning problems to concrete (hyper)graph rewriting problems. We conclude in Section 6 with a summary of directions for future research and development. The majority of the core CATEGORICA functionality presented within this article, and its forthcoming companion article, has been fully documented and exposed via the *Wolfram Function Repository*, including functions such as `AbstractQuiver`, `AbstractCategory`, `AbstractFunctor`, `AbstractProduct`, `AbstractCoproduct`, `AbstractPullback`, `AbstractPushout` and `AbstractStrictMonoidalCategory`. However, there is also a significant amount of functionality that has not yet been documented and/or exposed, but which is nevertheless available through the CATEGORICA GitHub Repository (with the overall framework currently consisting of over 20,000 lines of symbolic, high-level Wolfram Language code). All explicit examples of categories $\mathcal{C}$ presented within this article are *small*, in the sense that the object sets $\mathrm{ob}(\mathcal{C})$ and morphism sets $\mathrm{hom}(\mathcal{C})$ are both actually sets, rather than proper classes. Indeed, the only categories currently supported explicitly by CATEGORICA are strictly *finite*, which enables us to bypass any considerations of set-theoretic foundations.

2. Quivers, Categories and Diagrams

Every category represented within the CATEGORICA framework has, as its underlying "skeleton", a directed multigraph called a *quiver* (i.e. a collection of vertices called *objects*, connected by directed edges called *arrows*, which can have arbitrary multiplicity). One important terminological convention that is enforced consistently throughout the CATEGORICA framework is that the directed edges in a quiver are always known as *arrows*, whereas the directed edges in the corresponding category that is generated by that quiver (potentially freely, potentially with additional algebraic structure) are always known as *morphisms*. By and large, most conventions regarding the notation and nomenclature used within the CATEGORICA framework have been chosen to reflect the conventions set within Mac Lane's classic treatise on the subject [38]. If $\mathcal{Q}$ is an arbitrary quiver, then we shall denote the set of objects/nodes in $\mathcal{Q}$ by $\mathrm{ob}(\mathcal{Q})$, and the set of directed edges/arrows in $\mathcal{Q}$ by $\mathrm{arr}(\mathcal{Q})$. The rules for how to generate a category $\mathcal{C}$ (whose set of objects is denoted $\mathrm{ob}(\mathcal{C})$ and whose set of morphisms is denoted $\mathrm{hom}(\mathcal{C})$, by analogy to sets of homomorphisms in abstract algebra) from the underlying quiver $\mathcal{Q}$ are then very straightforward: firstly, for every object X or arrow $f : X \to Y$ in the quiver $\mathcal{Q}$, there is a corresponding object X or morphism $f : X \to Y$ in the category $\mathcal{C}$:

$$\forall X \in \mathrm{ob}(\mathcal{Q}), \qquad \exists X \in \mathrm{ob}(\mathcal{C}), \qquad \text{and} \qquad \forall (f : X \to Y) \in \mathrm{arr}(\mathcal{Q}), \qquad \exists (f : X \to Y) \in \mathrm{hom}(\mathcal{C}); \tag{1}$$

 doi:10.20944/preprints202403.1479.v1

secondly, for every object X in the quiver $\mathcal{Q}$, there exists a corresponding morphism $id_X : X \to X$ in the category $\mathcal{C}$ mapping that object to itself:

$$\forall X \in \mathrm{ob}(\mathcal{Q}), \qquad \exists (id_X : X \to X) \in \mathrm{hom}(\mathcal{C}); \tag{2}$$

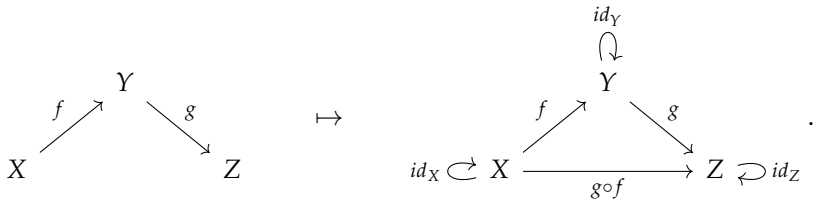
and, thirdly, for every pair of morphisms $f : X \to Y$ and $g : Y \to Z$ in the category $\mathcal{C}$ where the *codomain* of the first morphism matches the *domain* of the second, i.e. $\mathrm{codom}(f : X \to Y) = \mathrm{dom}(g : Y \to Z)$ (in this case, for instance, the domain of the arrow/morphism $f : X \to Y$ is X, i.e. $\mathrm{dom}(f : X \to Y) = X$, and its codomain is Y, i.e. $\mathrm{codom}(f : X \to Y) = Y$), there exists a third morphism $g \circ f : X \to Z$ mapping from the domain of the first morphism to the codomain of the second:

$$\forall (f : X \to Y), (g : Y \to Z) \in \mathrm{hom}(\mathcal{C}), \qquad \exists (g \circ f : X \to Z) \in \mathrm{hom}(\mathcal{C}), \tag{3}$$

or, slightly more abstractly (omitting the arbitrary object labels X, Y and Z):

$$\forall f, g \in \mathrm{hom}(\mathcal{C}), \qquad \text{such that} \qquad \mathrm{codom}(f) = \mathrm{dom}(g), \\ \exists (g \circ f : \mathrm{dom}(f) \to \mathrm{codom}(g)) \in \mathrm{hom}(\mathcal{C}). \tag{4}$$

The resulting category $\mathcal{C}$ is known as the *free category* generated by the quiver $\mathcal{Q}$ (with the arrows of $\mathcal{Q}$ acting as the generators, in the algebraic sense, of the morphisms of $\mathcal{C}$). This construction can be illustrated diagrammatically by means of the following minimal example:



. (5)

As shown in Figure 1, the `AbstractQuiver` and `AbstractCategory` functions in CATEGORICA allow one to represent arbitrary (finite) quivers and categories within the language, with both quivers and free categories being represented internally as lists of objects and associations of arrows/morphisms between those objects (which are then trivially interconvertible with the corresponding combinatorial representations of these structures as labeled directed graphs); every `AbstractCategory` object comes equipped with an associated `AbstractQuiver` object representing its underlying "skeletal" structure.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 March 2024 doi:10.20944/preprints202403.1479.v1

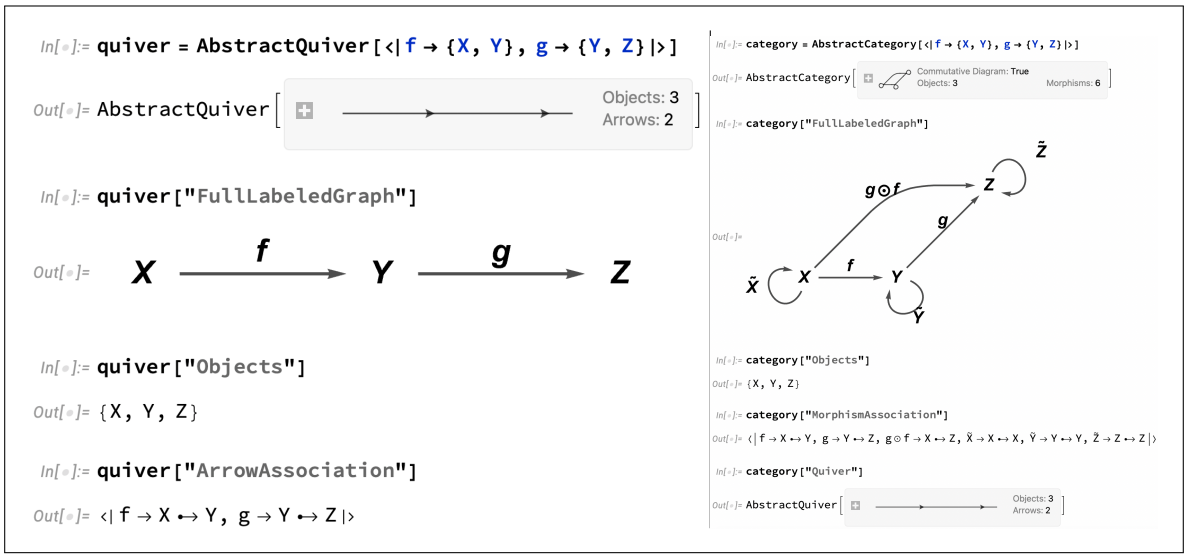


Figure 1. On the left, the `AbstractQuiver` object for a simple (three-object, two-arrow) quiver. On the right, the `AbstractCategory` object for the three-object, six-morphism category that is freely generated by this quiver.

Within the construction of the free category $\mathcal{C}$ described above, the morphisms $id_X : X \to X$ that were introduced for each object X are known as *identity* morphisms on X, and the abstract binary operation:

$$\circ : \hom(\mathcal{C}) \times \hom(\mathcal{C}) \to \hom(\mathcal{C}), \tag{6}$$

that appears in the definition of the morphism $g \circ f : X \to Z$ is known as the *composition* operation (such that the resulting morphism $g \circ f : X \to Z$ may be referred to as the *composition* of morphisms $f : X \to Y$ and $g : Y \to Z$). The identity morphisms $id_X : X \to X$ must satisfy the requisite axioms to act as both left and right identities under the operation of morphism composition, such that if $f : X \to Y$ is a morphism in the category $\mathcal{C}$, then $f \circ id_X : X \to Y$ should be the same as $f : X \to Y$ (i.e. id_X acts as a right identity on f):

$$\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f \circ id_X : X \to Y) = (f : X \to Y), \tag{7}$$

or, illustrated diagrammatically:

$$id_X \circlearrowright X \underset{f \circ id_X}{\overset{f}{\rightrightarrows}} Y \qquad \mapsto \qquad id_X \circlearrowright X \xrightarrow{f = f \circ id_X} Y \ , \tag{8}$$

and, similarly, $id_Y \circ f : X \to Y$ should be the same as $f : X \to Y$ (i.e. id_Y acts as a left identity on f):

$$\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (id_Y \circ f : X \to Y) = (f : X \to Y), \tag{9}$$

or, illustrated diagrammatically:

$$X \underset{id_Y \circ f}{\overset{f}{\rightrightarrows}} Y \circlearrowright id_Y \qquad \mapsto \qquad X \xrightarrow{f = id_Y \circ f} Y \circlearrowright id_Y \ . \tag{10}$$

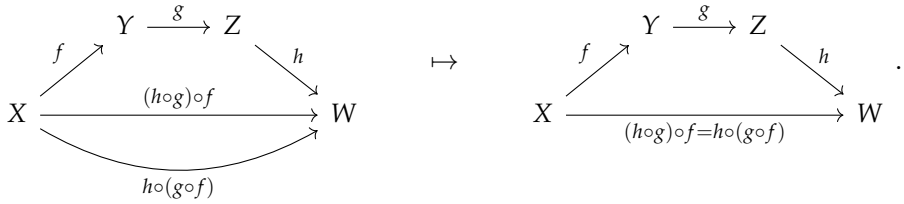
Additionally, the operation of morphism composition itself must satisfy the axiom of associativity, such that if $f : X \to Y$, $g : Y \to Z$ and $h : Z \to W$ form a set of three composable morphisms in

 doi:10.20944/preprints202403.1479.v1

the category $\mathcal{C}$, then the composition $(h \circ g) \circ f : X \to W$ should be the same as the composition $h \circ (g \circ f) : X \to W$:

$$\forall (f : X \to Y), (g : Y \to Z), (h : Z \to W) \in \hom(\mathcal{C}),$$
$$((h \circ g) \circ f : X \to W) = (h \circ (g \circ f) : X \to W), \quad (11)$$

or, illustrated diagrammatically:



(12)

CATEGORICA automatically keeps track of all algebraic equivalences between morphisms that must be imposed in order to maintain consistency with these identity and associativity axioms, as shown in Figure 2 for the case of two relatively simple `AbstractCategory` objects. Note that the `CircleDot[ ... ]` and `OverTilde[ ... ]` representations of the composition operation $\circ$ and the identity morphisms id_X, respectively, are simply the defaults chosen by CATEGORICA, and can be overridden simply by passing additional arguments to `AbstractCategory`.

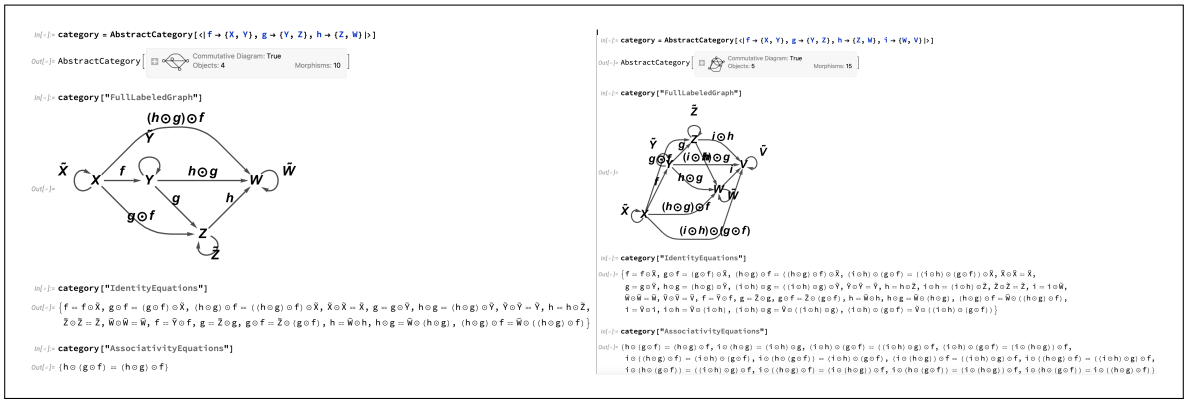
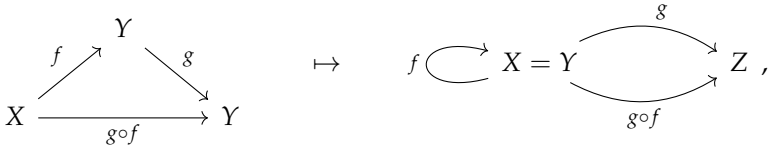


Figure 2. On the left, the `AbstractCategory` object for a simple (four-object, ten-morphism) category, showing all algebraic equivalences between morphisms that must hold by virtue of the identity and associativity axioms of category theory. On the right, the `AbstractCategory` object for a slightly larger (five-object, fifteen-morphism) category, again showing all algebraic equivalences between morphisms that must hold by virtue of the identity and associativity axioms of category theory.

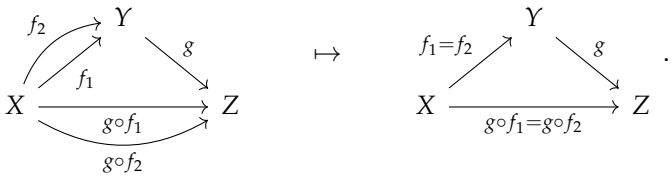
Although we have considered only *free* categories so far, it is also possible to construct non-free categories which possess additional algebraic structure in the form of additional equivalences between their objects and morphisms (beyond simply the minimal algebraic equivalences necessitated by the axioms of category theory). For instance, we could consider imposing the equivalence $X = Y$ between objects X and Y in the following simple category:



(13)

 doi:10.20944/preprints202403.1479.v1

or the equivalence $(f_1 : X \to Y) = (f_2 : X \to Y)$ between morphisms $f_1 : X \to Y$ and $f_2 : X \to Y$ in the following, slightly different, simple category:



(14)

Note that we are only permitted to impose algebraic equivalences between morphisms whose domains and codomains both match, i.e., abstractly, using the functions $\mathrm{dom} : \mathrm{hom}(\mathcal{C}) \to \mathrm{ob}(\mathcal{C})$ and $\mathrm{codom} : \mathrm{hom}(\mathcal{C}) \to \mathrm{ob}(\mathcal{C})$ introduced above, we have:

$$\forall f, g \in \mathrm{hom}(\mathcal{C}), \qquad \Longrightarrow \qquad \mathrm{dom}(f) = \mathrm{dom}(g), \qquad \text{and} \qquad \mathrm{codom}(f) = \mathrm{codom}(g). \tag{15}$$

Note also that an algebraic equivalence between morphisms, such as $(f_1 : X \to Y) = (f_2 : X \to Y)$ in the above, will, in general, automatically imply other algebraic equivalences between morphisms, such as:

$$(g \circ f_1 : X \to Z) = (g \circ f_2 : X \to Z), \tag{16}$$

as shown in the corresponding diagram, although in this particular case the converse does not necessarily hold, as we shall see in more detail later. In Figure 3, we see the `AbstractCategory` objects corresponding to the two examples presented above (with object equivalence $X = Y$ and morphism equivalence $(f_1 : X \to Y) = (f_2 : X \to Y)$ specified, respectively), showing the directed graph representations of the corresponding categories both with and without algebraic equivalences imposed. One of the general design principles of CATEGORICA is that the keyword *"Reduced"* within property names such as *"ReducedLabeledGraph"* is used to indicate that all known algebraic equivalences should be applied (as opposed to the keyword *"Full"* within property names such as *"FullLabeledGraph"*, which is used to indicate that no algebraic equivalences should be applied, and that the full, "maximally-unreduced" algebraic structure should be presented instead). There is also a related keyword *"Simple"* (as in the property name *"SimpleLabeledGraph"*), which can be used to remove all self-loops and multi-edges from the corresponding directed graph representation of any quiver, category, functor, etc. (for instance in order to yield a more presentable form of a particular diagrammatic representation, of the kind that might be shown within an academic paper). Although this automatic simplification feature is frequently very useful when performing practical diagrammatic manipulations, since the purpose of the present article is to be completely explicit and to illustrate (in as much detail as necessary) the underlying design and algorithmic underpinnings of the CATEGORICA framework, we shall not make use of it here.

 doi:10.20944/preprints202403.1479.v1

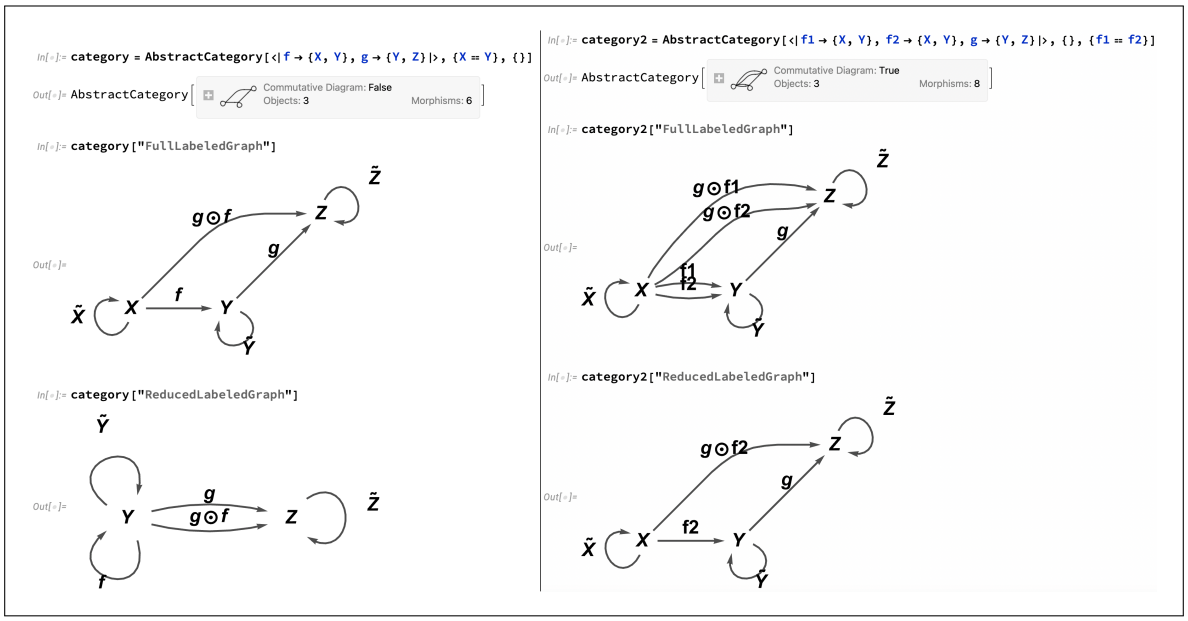


Figure 3. On the left, the `AbstractCategory` object for a simple category with the object equivalence $X = Y$ specified, showing the labeled graph representations of the category both with and without algebraic equivalences imposed. On the right, the `AbstractCategory` object for a simple category with the morphism equivalence $(f_1 : X \to Y) = (f_2 : X \to Y)$ specified, showing the labeled graph representations of the category both with and without algebraic equivalences imposed.

One of the most common applications of non-free categories and algebraic equivalences between morphisms, occurring throughout many fields of mathematics (and especially in abstract and homological algebra), is the formal treatment of *commutative diagrams*. Indeed, it is often said that commutative diagrams play the same role in category theory and abstract/homological algebra that equations play in more elementary algebra. For instance, when one says that the following diagram *commutes*:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle i} \\ Z & \xrightarrow[h]{} & W \end{array}\ , \tag{17}$$

one is effectively imposing the following equivalence between morphisms:

$$(h \circ g : X \to W) = (i \circ f : X \to W), \tag{18}$$

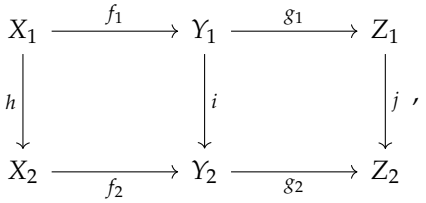
or, in diagrammatic form, one has:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle g}\downarrow & \overset{h\circ g}{\searrow} & \downarrow{\scriptstyle i} \\ Z & \xrightarrow[h]{} & W \end{array} \quad \overset{i\circ f}{\longmapsto} \quad \begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle g}\downarrow & \overset{h\circ g = i\circ f}{\searrow} & \downarrow{\scriptstyle i} \\ Z & \xrightarrow[h]{} & W \end{array}\ . \tag{19}$$

This condition of commutativity of diagrams may thus be characterized purely combinatorially, namely as the condition that all directed paths through the labeled graph representation of the associated category yield the same morphism (up to algebraic equivalence). CATEGORICA is hence able to fall back to using purely graph-theoretic search algorithms in order to compute the minimum set of morphism

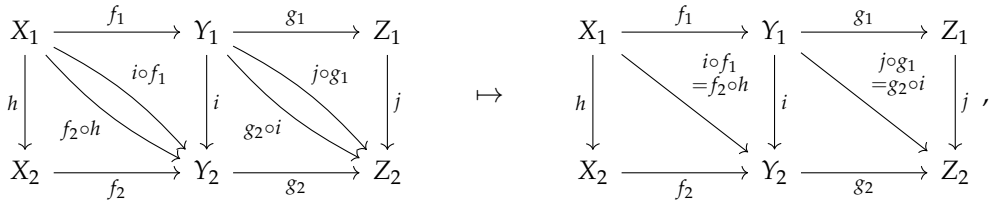
 doi:10.20944/preprints202403.1479.v1

equivalences necessary to force this diagram to commute, as well as to prove that these equivalences are indeed sufficient, as shown in Figure 4. However, although the case of a commutative square is very straightforward, it does not take long before this method of *diagram chasing* becomes essentially unmanageable for a human algebraist to enact in full detail; for instance, even upon considering the next obvious case, namely a commutative *oblong* of the form:



(20)

things have already become quite a bit trickier to analyze than they were for the commutative square, since one not only needs to force the two interior squares to commute:

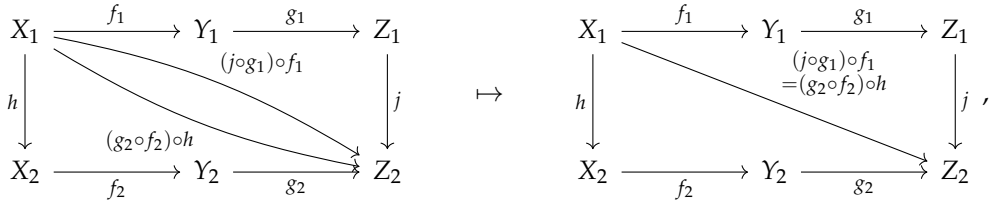


(21)

i.e:

$$(i \circ f_1 : X_1 \to Y_2) = (f_2 \circ h : X_1 \to Y_2), \qquad \text{and} \qquad (j \circ g_1 : Y_1 \to Z_2) = (g_2 \circ i : Y_1 \to Z_2), \tag{22}$$

but one must also somehow take care of the commutativity of the outer rectangle:



(23)

i.e:

$$((j \circ g_1) \circ f_1 : X_1 \to Z_2) = ((g_2 \circ f_2) \circ h : X_1 \to Z_2). \tag{24}$$

Nevertheless, despite the additional complexity, CATEGORICA is able to handle this case (and, indeed, the case of much larger and more complicated diagrams) in much the same way, as demonstrated in Figure 5. A fully rigorous definition and treatment of categorical diagrams necessitates the introduction of a certain functor defined over *index categories* (or *schemes*), which we shall revisit following our introduction to CATEGORICA's handling of abstract functor objects in Section 4. Note that `AbstractQuiver` objects in CATEGORICA may also carry algebraic equivalence information on both objects and arrows: these equivalences are then translated into corresponding algebraic equivalences on objects and *morphisms* upon promotion of the quiver to a full `AbstractCategory` object via the pipeline outlined above.

 doi:10.20944/preprints202403.1479.v1

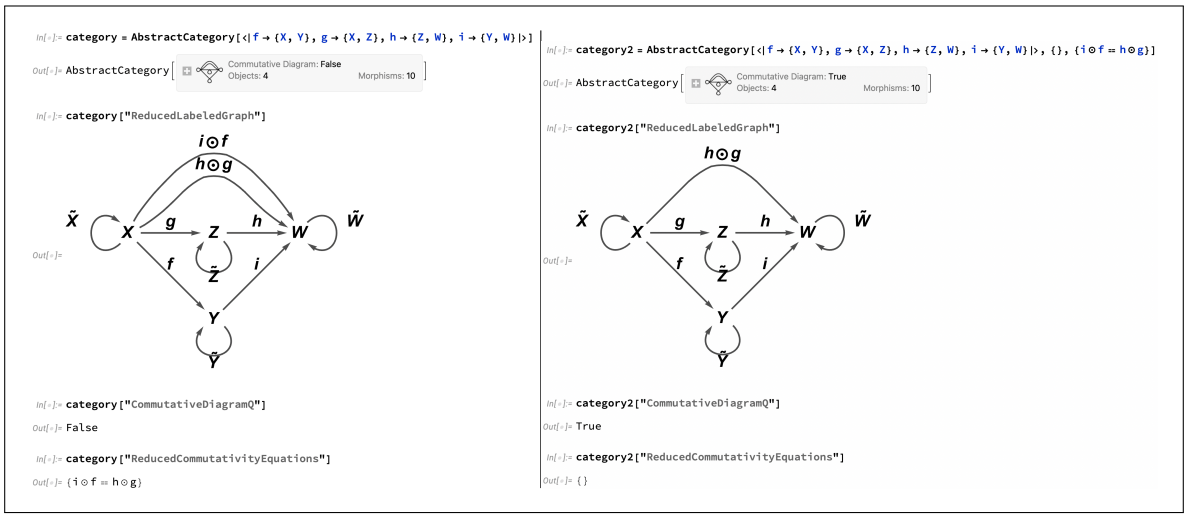


Figure 4. On the left, the `AbstractCategory` object corresponding to a simple (not yet commutative) square diagram, illustrating that the diagram is not commutative and showing that the morphism equivalence $i \circ f = h \circ g$ is the minimal algebraic condition necessary to force the diagram to commute. On the right, the `AbstractCategory` object for the commutative case of this same square diagram, with the morphism equivalence $i \circ f = h \circ g$ imposed, demonstrating that this equivalence is indeed sufficient to force the diagram to commute.

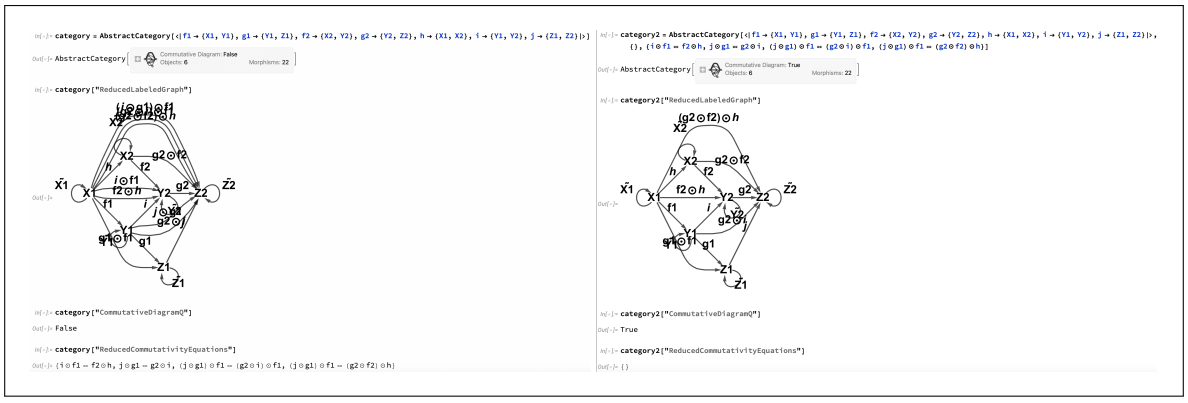


Figure 5. On the left, the `AbstractCategory` object corresponding to a slightly more complex (not yet commutative) oblong diagram, illustrating that the diagram is not yet commutative and computing the minimum set of morphism equivalences necessary to force the diagram to commute. On the right, the `AbstractCategory` object for the commutative case of this same oblong diagram, with the aforementioned morphism equivalences imposed, demonstrating that these equivalences are indeed sufficient to force the diagram to commute.

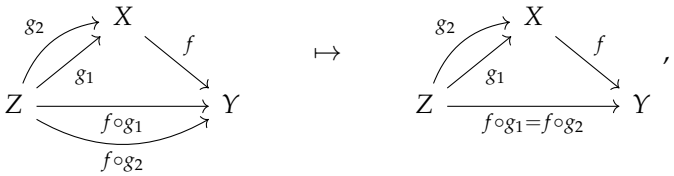
 doi:10.20944/preprints202403.1479.v1

3. Monos, Epis, Retractions and Sections: The Case of Groupoids

The category-theoretic analog of an injective function in mathematical analysis is a *monomorphism* [39,40]: a left-cancellative morphism. Specifically, if the morphism $f : X \to Y$ in the category $\mathcal{C}$ is such that, for all objects Z and all pairs of morphisms $g_1 : Z \to X$ and $g_2 : Z \to X$ such that $(f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y)$, one is able to "cancel on the left" by $f : X \to Y$ to obtain that $(g_1 : Z \to X) = (g_2 : Z \to X)$, then $f : X \to Y$ is a monomorphism:

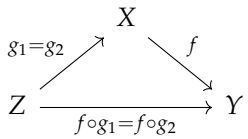
$$\begin{aligned} &\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f : X \to Y) \text{ is a monomorphism} \\ &\qquad \iff \qquad \forall Z \in \mathrm{ob}(\mathcal{C}), \qquad \forall (g_1 : Z \to X), (g_2 : Z \to X) \in \hom(\mathcal{C}), \\ &(f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y) \qquad \implies \qquad (g_1 : Z \to X) = (g_2 : Z \to X), \end{aligned} \tag{25}$$

or, illustrated diagrammatically, one has that if:



(26)

then one also necessarily has:

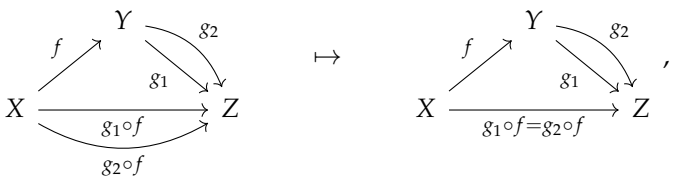


(27)

The corresponding *dual* notion to that of a monomorphism (i.e. the construction obtained by reversing the direction of the morphisms, and hence reversing the order of morphism composition, in the diagrams shown above) is that of an *epimorphism*: a right-cancellative morphism, and hence the category-theoretic analog of a surjective function in analysis. Specifically, if the morphism $f : X \to Y$ in the category $\mathcal{C}$ is such that, for all objects Z and all pairs of morphisms $g_1 : Y \to Z$ and $g_2 : Y \to Z$ such that $(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z)$, one is able to "cancel on the right" by $f : X \to Y$ to obtain that $(g_1 : Y \to Z) = (g_2 : Y \to Z)$, then $f : X \to Y$ is an epimorphism:

$$\begin{aligned} &\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f : X \to Y) \text{ is an epimorphism} \\ &\qquad \iff \qquad \forall Z \in \mathrm{ob}(\mathcal{C}), \qquad \forall (g_1 : Y \to Z), (g_2 : Y \to Z) \in \hom(\mathcal{C}), \\ &(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z) \qquad \implies \qquad (g_1 : Y \to Z) = (g_2 : Y \to Z), \end{aligned} \tag{28}$$

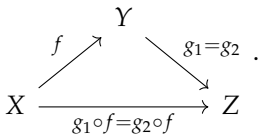
or, illustrated diagrammatically, one has that if:



(29)

then one also necessarily has:

 doi:10.20944/preprints202403.1479.v1



(30)

Figure 6 illustrates the basic diagrammatic setup for both monomorphisms and epimorphisms (commonly abbreviated simply to *monos* and *epis* in the category-theoretic literature) in CATEGORICA, and demonstrates in these two cases that all morphisms initially, i.e. in the absence of any further algebraic equivalences, correspond to monomorphisms and epimorphisms, respectively. Figure 7 shows that, by imposing the algebraic equivalence $(f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y)$ in the former case and $(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z)$ in the latter case, one is able to force the morphism $f : X \to Y$ to cease to be a monomorphism in the former example, and to cease to be an epimorphism in the latter example. Finally, Figure 8 demonstrates that one is able to restore the status of morphism $f : X \to Y$ as a monomorphism (in the former case) or an epimorphism (in the latter case) by imposing the additional algebraic equivalence $(g_1 : Z \to X) = (g_2 : Z \to X)$ (for the monomorphism case) or $(g_1 : Y \to Z) = (g_2 : Y \to Z)$ (for the epimorphism case). Note that any morphism that is both a monomorphism and an epimorphism is known as a *bimorphism* (the category-theoretic analog of a bijective function), and CATEGORICA contains in-built functionality for detecting and manipulating bimorphisms in much the same way.

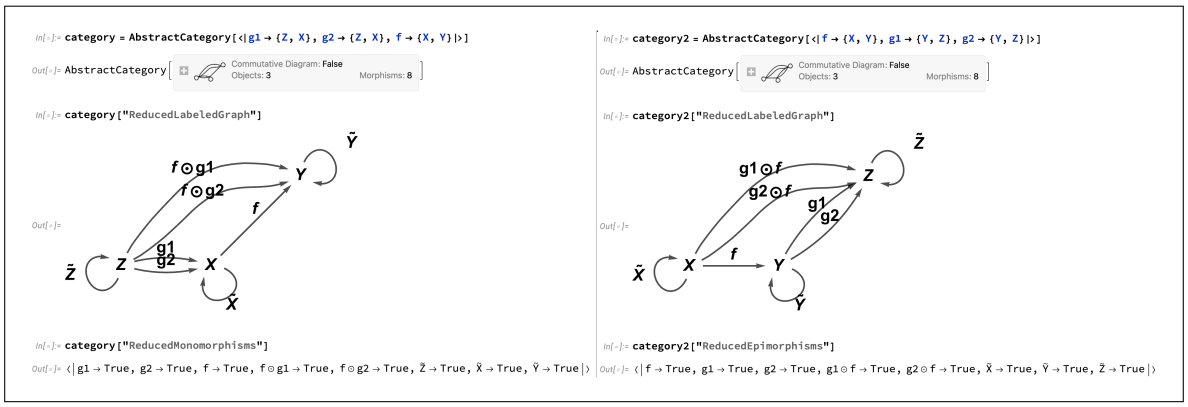


Figure 6. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a monomorphism, showing that all morphisms are initially monomorphisms. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of an epimorphism, showing that all morphisms are initially epimorphisms.

 doi:10.20944/preprints202403.1479.v1

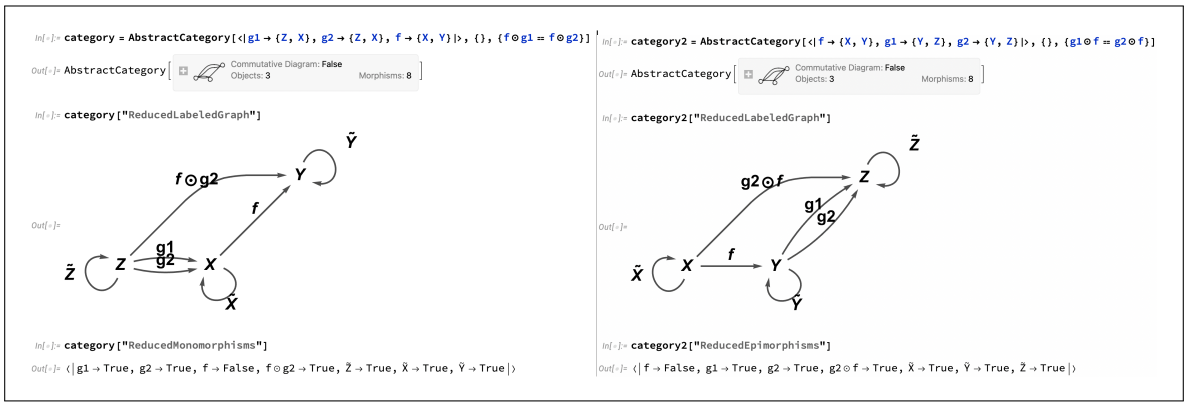


Figure 7. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a monomorphism, with the additional algebraic equivalence $f \circ g_1 = f \circ g_2$ imposed, showing that morphism f has now ceased to be a monomorphism. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of an epimorphism, with the additional algebraic equivalence $g_1 \circ f = g_2 \circ f$ imposed, showing that morphism f has now ceased to be an epimorphism.

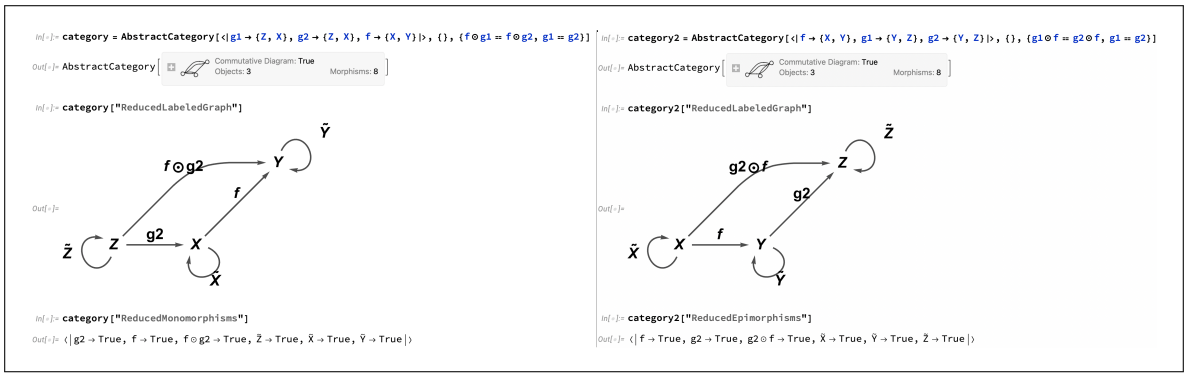


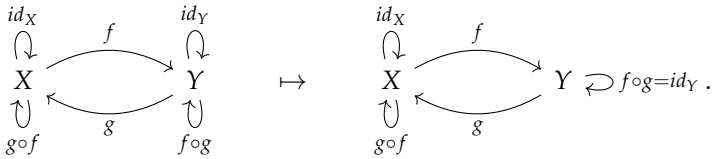
Figure 8. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a monomorphism, with the additional algebraic equivalences $f \circ g_1 = f \circ g_2$ *and* $g_1 = g_2$ both imposed, showing that morphism f has now returned to being a monomorphism. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of an epimorphism, with the additional algebraic equivalences $g_1 \circ f = g_2 \circ f$ *and* $g_1 = g_2$ both imposed, showing that morphism f has now returned to being an epimorphism.

An important special case of monomorphisms are *sections* [38,41]: morphisms that possess left inverses (and hence which are, themselves, right inverses of some other morphism). Likewise, an important special case of epimorphisms are *retractions*: dual to sections, these are morphisms that possess right inverses (and hence which are, themselves, left inverses of some other morphism). Specifically, if the morphisms $f : X \to Y$ and $g : Y \to X$ in the category $\mathcal{C}$ are such that $f \circ g : Y \to Y$ is the identity morphism on Y (i.e. $(f \circ g : Y \to Y) = (id_Y : Y \to Y)$), then $f : X \to Y$ is a retraction of $g : Y \to X$, and $g : Y \to X$ is a section of $f : X \to Y$:

$$\begin{aligned} &\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f : X \to Y) \text{ is a retraction} \\ &\quad \iff \qquad \exists (g : Y \to X) \in \hom(\mathcal{C}), \qquad \text{such that} \qquad (f \circ g : Y \to Y) = (id_Y : Y \to Y), \end{aligned} \tag{31}$$

or, illustrated diagrammatically, a retraction is a morphism $f : X \to Y$ such that one has:

 doi:10.20944/preprints202403.1479.v1

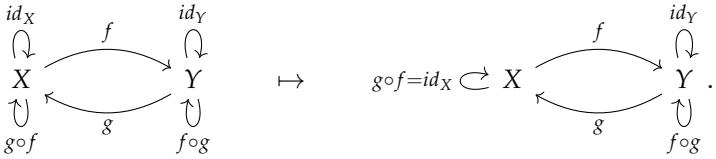


(32)

Similarly, if $g \circ f : X \to X$ is the identity morphism on X (i.e. $(g \circ f : X \to X) = (id_X : X \to X)$) then $f : X \to Y$ is a section of $g : Y \to X$, and $g : Y \to X$ is a retraction of $f : X \to Y$:

$$\begin{aligned} &\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f : X \to Y) \text{ is a section} \\ &\quad \iff \qquad \exists (g : Y \to X) \in \hom(\mathcal{C}), \qquad \text{such that} \qquad (g \circ f : X \to X) = (id_X : X \to X), \end{aligned} \tag{33}$$

or, illustrated diagrammatically, a section is a morphism $f : X \to Y$ such that one has:



(34)

These category-theoretic notions of retractions and sections are so-named because they naturally generalize the corresponding notions in topology, wherein one considers *retractions* of topological spaces into subspaces to be continuous maps that preserve all points in the subspace (and for which the corresponding inclusion maps of the subspaces into the original spaces, such that the compositions of the two maps always reduce to the identity maps on the subspaces, would be *sections*) [42,43]. Figure 9 illustrates the basic diagrammatic setup for both retractions and sections in CATEGORICA, and demonstrates that initially (in the absence of any further algebraic equivalences) no morphisms, with the exception of identity morphisms, are either retractions or sections, and therefore that no morphisms, with the exception of identity morphisms, possess either left or right inverses. However, by imposing the algebraic equivalence $(f \circ g : Y \to Y) = (id_Y : Y \to Y)$, one is able to force the morphism $f : X \to Y$ to be a retraction (and hence to possess the morphism $g : Y \to X$ as its right inverse), as well as to force the morphism $g : Y \to X$ to be a section (and hence to possess the morphism $f : X \to Y$ as its left inverse). It is therefore straightforward to prove, using CATEGORICA's diagrammatic theorem-proving capabilities, that every section is necessarily a monomorphism (i.e. that the existence of a left inverse necessarily implies left-cancellativity) and that every retraction is necessarily an epimorphism (i.e. that the existence of a right inverse necessarily implies right-cancellativity).

 doi:10.20944/preprints202403.1479.v1

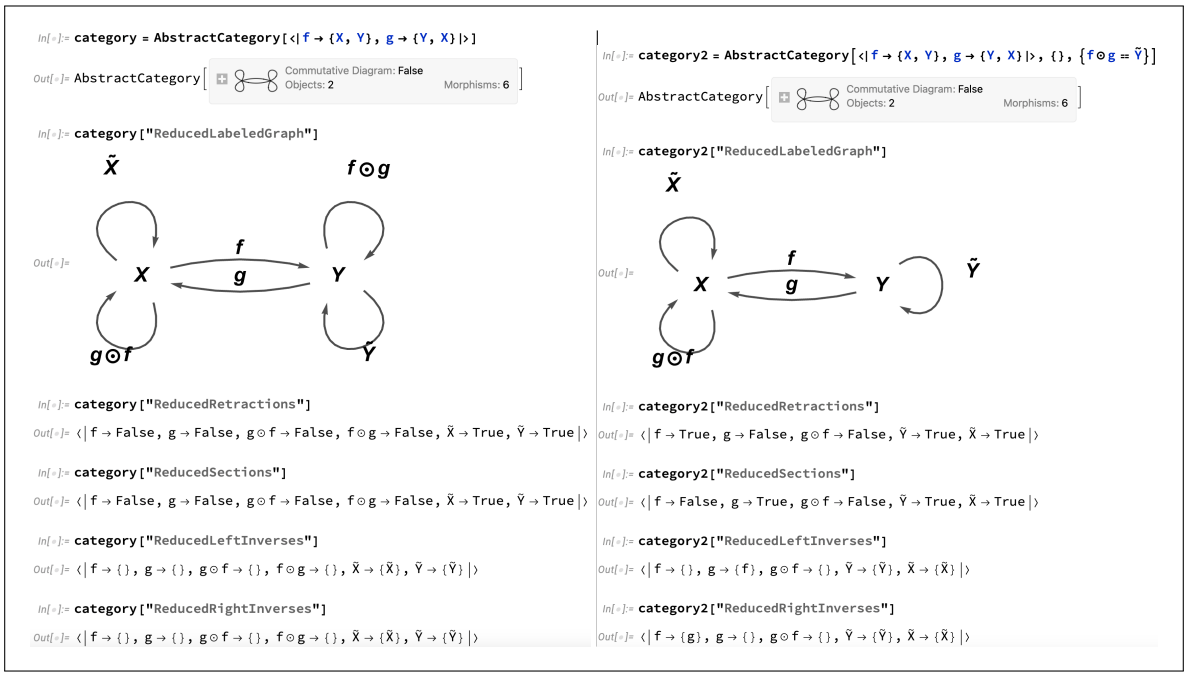
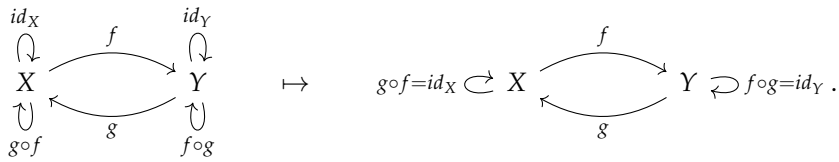


Figure 9. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a retraction/section, showing that initially no morphisms (except for identity morphisms) are either retractions or sections, and therefore that no morphisms (except for identity morphisms) possess either left or right inverses. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a retraction/section, with the additional algebraic equivalence $f \circ g = id_Y$ imposed, showing that morphism f has now become a retraction, and morphism g has now become a section, with f being the left inverse of g and g being the right inverse of f.

A morphism that acts as both a retraction and a section (and therefore which both is, and also possesses, a left and right inverse) is known as an *isomorphism*. Specifically, if the morphism $f : X \to Y$ in the category $\mathcal{C}$ is such that there exists another morphism $g : Y \to X$ such that both $g \circ f : X \to X$ is the identity morphism on X (i.e. $(g \circ f : X \to X) = (id_X : X \to X)$) and $f \circ g : Y \to Y$ is the identity morphism on Y (i.e. $(f \circ g : Y \to Y) = (id_Y : Y \to Y)$, then $f : X \to Y$ is an isomorphism (as, correspondingly, is $g : Y \to X$):

$$\begin{aligned} &\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad (f : X \to Y) \text{ is an isomorphism} \\ &\quad \iff \qquad \exists (g : Y \to X) \in \hom(\mathcal{C}), \qquad \text{such that} \qquad (g \circ f : X \to X) = (id_X : X \to X) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{and} \qquad (f \circ g : Y \to Y) = (id_Y : Y \to Y), \end{aligned} \tag{35}$$

or, illustrated diagrammatically, an isomorphism is a morphism $f : X \to Y$ such that one has:

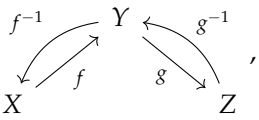


(36)

A category in which all morphisms are isomorphisms is known as a *groupoid*, since the morphisms of a groupoid consisting of a single object trivially form a group under the operation of morphism composition [44] (with the associativity and identity axioms deriving from the underlying axioms of category theory, and the inverse axiom deriving from the existence of both left and right inverses for

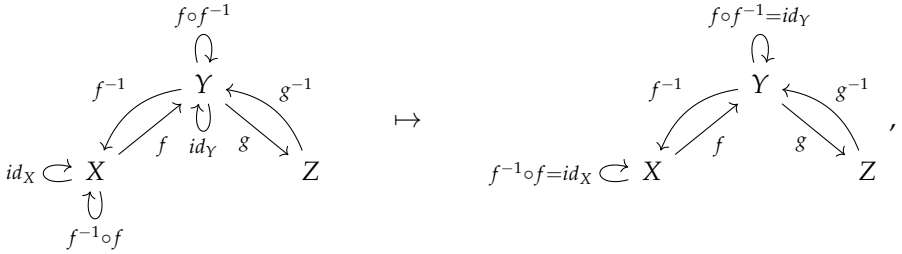
 doi:10.20944/preprints202403.1479.v1

each morphism), and therefore the morphisms of a groupoid consisting of multiple objects may be thought of as forming an abstract generalization of a group in which the group operation is *partial*, i.e. only well-defined for particular pairs of elements. Groupoids are particularly widely-studied in algebraic topology and homotopy theory, wherein the *fundamental groupoid* of a topological space generalizes the more traditional fundamental group to the case where one does not necessarily fix a single distinguished base point [45]. Much like in the case of commutative diagrams discussed previously, CATEGORICA is able to use purely graph-theoretic algorithms to compute the minimum set of morphism equivalences necessary to force the category shown above to be a groupoid, as well as to prove that these equivalences are indeed sufficient to do so, as shown in Figure 10. However, once the initial category is even marginally more complex than this, the task of determining the minimum set of algebraic conditions necessary for the category to be groupoidal quickly evolves to be highly non-trivial; for instance, for the case of a three-object category of the form:



, (37)

one must consider not only the conditions necessary to force the morphisms $f : X \to Y$ and $g : Y \to Z$ (and hence also the morphisms $f^{-1} : Y \to X$ and $g^{-1} : Z \to Y$) to be isomorphisms, namely:

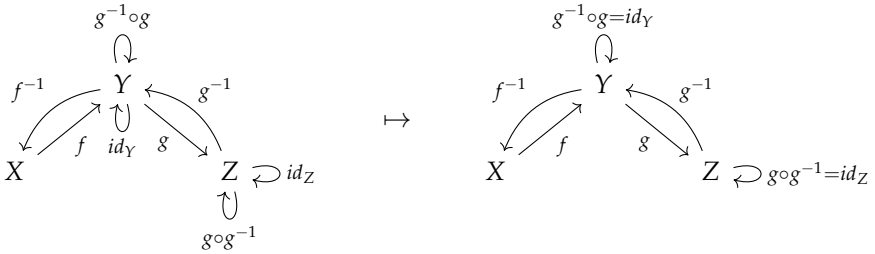


, (38)

i.e:

$$\left(f^{-1} \circ f : X \to X\right) = (id_X : X \to X), \qquad \text{and} \qquad \left(f \circ f^{-1} : Y \to Y\right) = (id_Y : Y \to Y), \tag{39}$$

and:



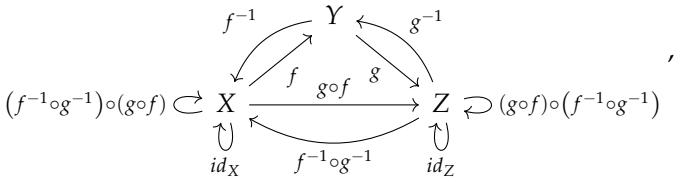
, (40)

i.e:

$$\left(g^{-1} \circ g : Y \to Y\right) = (id_Y : Y \to Y), \qquad \text{and} \qquad \left(g \circ g^{-1} : Z \to Z\right) = (id_Z : Z \to Z), \tag{41}$$

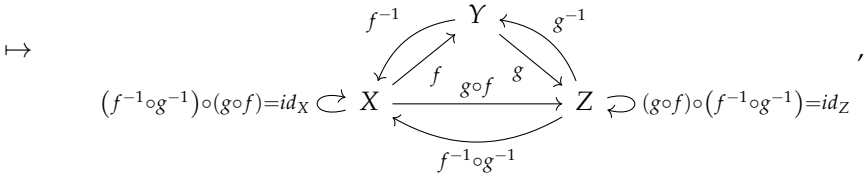
respectively, but also the conditions necessary to force the resulting pair of composite morphisms $g \circ f : X \to Z$ and $f^{-1} \circ g^{-1} : Z \to X$ to be isomorphisms as well, namely to collapse:

 doi:10.20944/preprints202403.1479.v1



(42)

down to:



(43)

in addition to any relevant permutations thereof. Nevertheless, the generality of CATEGORICA's algebraic reasoning algorithms ensures that it is able to handle such cases in exactly the same way, as demonstrated in Figure 11.

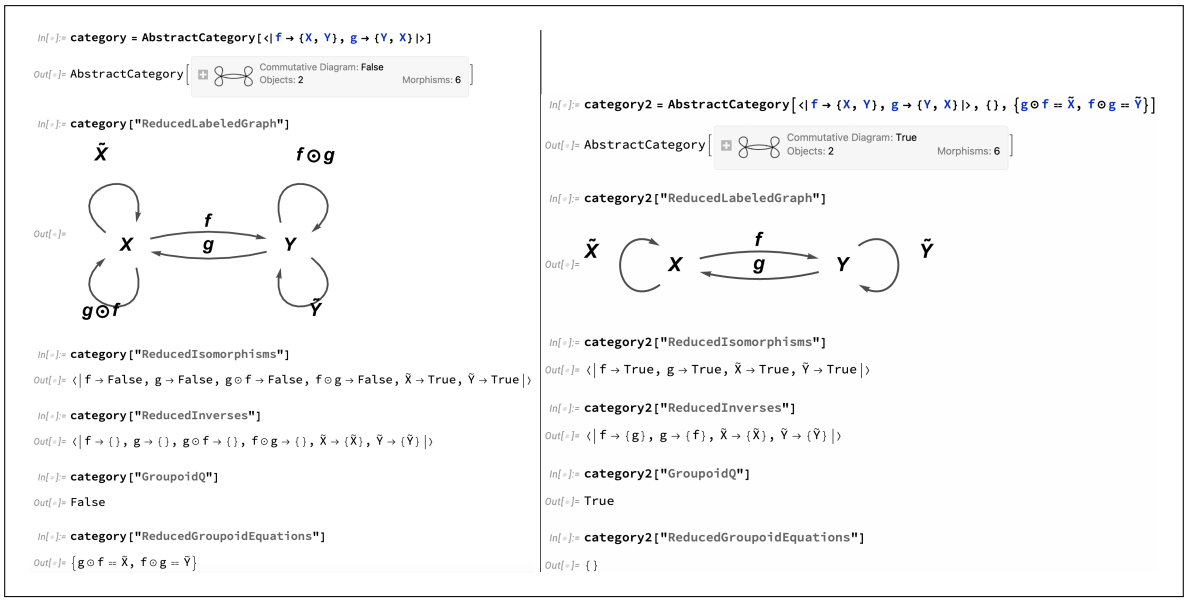


Figure 10. On the left, the `AbstractCategory` object corresponding to a simple (not yet groupoidal) category, showing that initially no morphisms (except for identity morphisms) are isomorphisms, and therefore that no morphisms (except for identity isomorphisms) possess inverses, hence illustrating that the category is not a groupoid and showing that the morphism equivalences $g \circ f = id_X$ and $f \circ g = id_Y$ are the minimal algebraic conditions necessary to force the category to be groupoidal. On the right, the `AbstractCategory` object for the groupoidal case of the same category, with the morphism equivalences $g \circ f = id_X$ and $f \circ g = id_Y$ imposed, showing that the morphisms f and g have now become isomorphisms, with f being the inverse of g and g being the inverse of f, hence demonstrating that these equivalences are indeed sufficient to force the category to be groupoidal.

 doi:10.20944/preprints202403.1479.v1

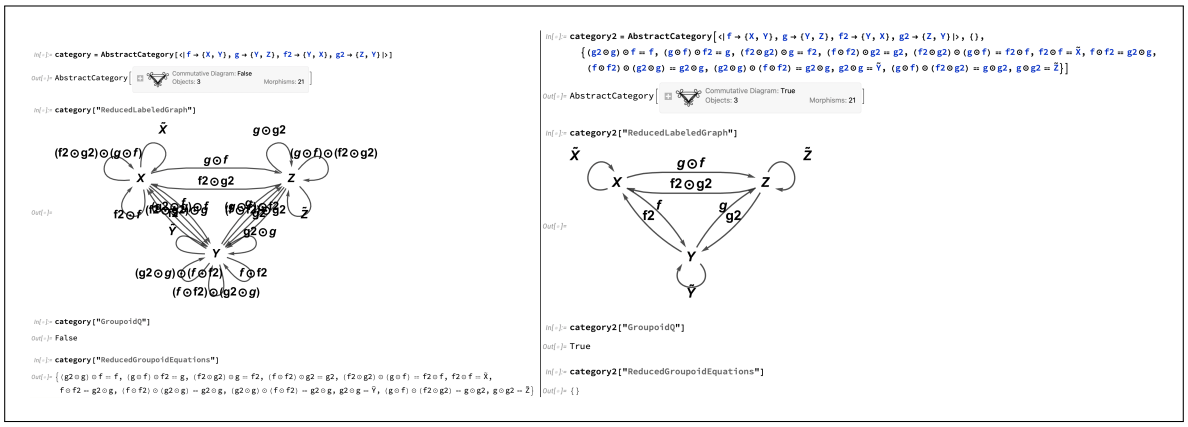


Figure 11. On the left, the `AbstractCategory` object corresponding to a slightly more complex (not yet groupoidal) category, illustrating that the category is not a groupoid and computing the minimum set of morphism equivalences necessary to force the category to be groupoidal. On the right, the `AbstractCategory` object for the groupoidal case of the same category, with the aforementioned morphism equivalences imposed, demonstrating that these equivalences are indeed sufficient to force the category to be groupoidal.

Dual constructions of this general kind, such as monomorphisms vs. epimorphisms, retractions vs. sections, etc., are ubiquitous throughout category theory, and can be investigated systematically using the *"DualCategory"* property of `AbstractCategory` objects in the CATEGORICA framework (e.g. a morphism that registers as a monomorphism within a particular `AbstractCategory` object will register as an epimorphism in the corresponding `AbstractCategory` object returned by the *"DualCategory"* property, etc.). A very straightforward example of a dual construction is that of *initial* vs. *terminal* objects; an initial object X in the category $\mathcal{C}$ is an object such that, for every object P (including X itself) in $\mathcal{C}$, there exists a unique *outgoing* morphism $f : X \to P$:

$$\forall X \in \mathrm{ob}(\mathcal{C}), \quad X \text{ is an initial object} \quad \Longleftrightarrow \quad \forall P \in \mathrm{ob}(\mathcal{C}), \quad \exists!(f : X \to P) \in \mathrm{hom}(\mathcal{C}), \tag{44}$$

or, illustrated diagrammatically:

$$X \xrightarrow{\exists! f} \forall P \ . \tag{45}$$

Dually, a terminal object X in the category $\mathcal{C}$ is an object such that, for every object P (including X itself) in $\mathcal{C}$, there exists a unique *incoming* morphism $f : P \to X$:

$$\forall X \in \mathrm{ob}(\mathcal{C}), \quad X \text{ is a terminal object} \quad \Longleftrightarrow \quad \forall P \in \mathrm{ob}(\mathcal{C}), \quad \exists!(f : P \to X) \in \mathrm{hom}(\mathcal{C}), \tag{46}$$

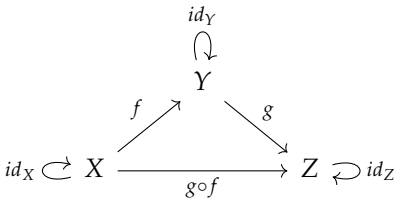
or, illustrated diagrammatically:

$$\forall P \xrightarrow{\exists! f} X \ . \tag{47}$$

Initial and terminal objects generalize many key construction in pure mathematics [38,46], such as the bottom/minimal and top/maximal elements $\bot$ and $\top$ of partially-ordered sets (since, if one considers a category $\mathcal{C}$ constructed from a given poset $\mathcal{P}$ whose objects are elements of $\mathcal{P}$ and where the morphism $f : X \to Y$ exists in $\mathcal{C}$ if and only if $X \leq Y$ in $\mathcal{P}$, then the bottom/minimal element $\bot$ is an initial object and the top/maximal element $\top$ is a terminal object in the category $\mathcal{C}$), and the empty and point

 doi:10.20944/preprints202403.1479.v1

spaces $\varnothing$ and $*$ in topology (since, if one considers the category **Top** whose objects are topological spaces and whose morphisms are continuous functions between them, then the empty space $\varnothing$ is an initial object in **Top** because there exists a unique continuous function mapping the empty space to any other topological space, and the point space $*$ is a terminal object in **Top** because there exists a unique continuous function mapping any topological space to the point space). If we take the simple triangular diagram case that we have investigated previously:

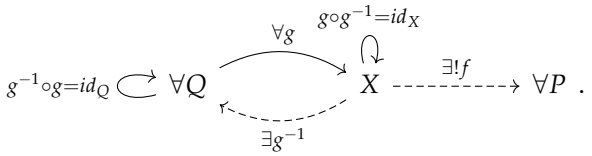


, (48)

then it is easy to see that X here is an initial object, and, dually, that Z is a final object, as illustrated by the elementary CATEGORICA implementation shown in Figure 12. However, a slightly more subtle refinement of the concept of an ordinary initial object is that of a *strict* initial object, namely an initial object X for which all incoming morphisms $f : Q \to X$ (where Q is an arbitrary object in the category $\mathcal{C}$) must be isomorphisms:

$$\begin{aligned}
&\forall X \in \mathrm{ob}(\mathcal{C}), \quad X \text{ is a strict initial object} \\
&\qquad \iff \quad \forall P \in \mathrm{ob}(\mathcal{C}), \quad \exists!(f : X \to P) \in \mathrm{hom}(\mathcal{C}), \quad \text{and} \\
&\forall Q \in \mathrm{ob}(\mathcal{C}), \quad \forall(g : Q \to X) \in \mathrm{hom}(\mathcal{C}), \quad \exists\left(g^{-1} : X \to Q\right) \in \mathrm{hom}(\mathcal{C}), \quad \text{such that,} \\
&\left(g \circ g^{-1} : X \to X\right) = (id_X : X \to X), \quad \text{and} \quad \left(g^{-1} \circ g : Q \to Q\right) = \left(id_Q : Q \to Q\right),
\end{aligned} \tag{49}$$

or, illustrated diagrammatically:

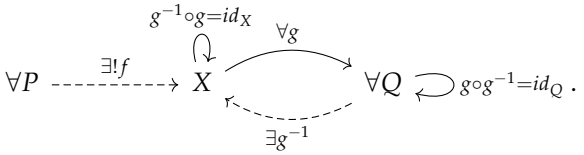


. (50)

Dually, a *strict* terminal object is a terminal object X for which all outgoing morphisms $f : X \to Q$ (where Q is again an arbitrary object in the category $\mathcal{C}$) must be isomorphisms:

$$\begin{aligned}
&\forall X \in \mathrm{ob}(\mathcal{C}), \quad X \text{ is a strict terminal object ,} \\
&\qquad \iff \quad \forall P \in \mathrm{ob}(\mathcal{C}), \quad \exists!(f : P \to X) \in \mathrm{hom}(\mathcal{C}), \quad \text{and} \\
&\forall Q \in \mathrm{ob}(\mathcal{C}), \quad \forall(g : X \to Q) \in \mathrm{hom}(\mathcal{C}), \quad \exists\left(g^{-1} : Q \to X\right) \in \mathrm{hom}(\mathcal{C}), \quad \text{such that} \\
&\left(g \circ g^{-1} : Q \to Q\right) = \left(id_Q : Q \to Q\right), \quad \text{and} \quad \left(g^{-1} \circ g : X \to X\right) = (id_X : X \to X),
\end{aligned} \tag{51}$$

or, illustrated diagrammatically:



. (52)

 doi:10.20944/preprints202403.1479.v1

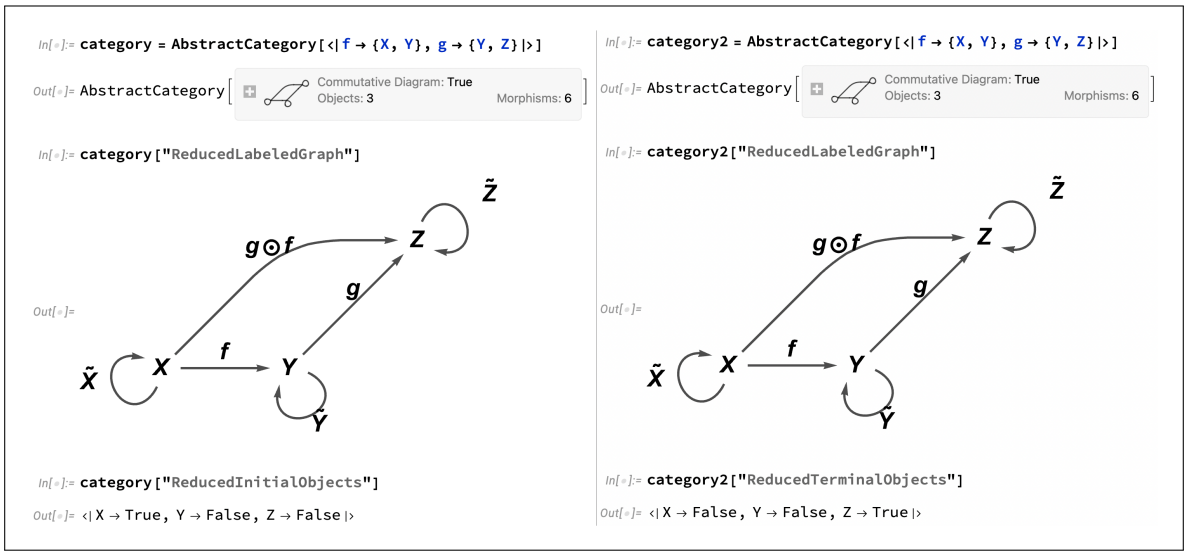
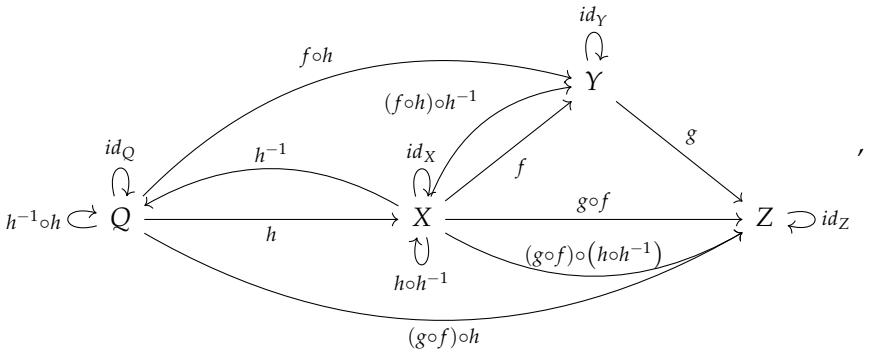


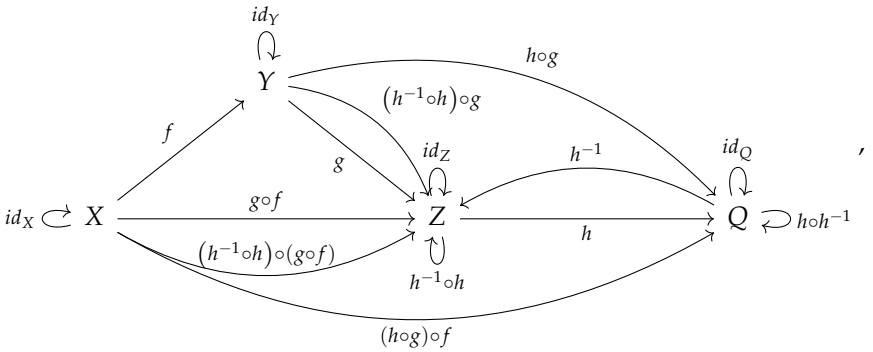
Figure 12. On the left, the `AbstractCategory` object corresponding to a simple triangular diagram, showing that object X is an initial object. On the right, the same `AbstractCategory` object corresponding to the same simple triangular diagram, showing that object Z is a terminal object.

Returning again to the simple triangular diagram above, and introducing either a new incoming morphism $h : Q \to X$ to the initial object X, or a new outgoing morphism $h : Z \to Q$ from the terminal object Z (along with corresponding morphisms $h^{-1} : X \to Q$ or $h^{-1} : Q \to Z$ in the reverse directions, respectively), i.e:



, (53)

or:



, (54)

respectively, we see that, in the first case, X ceases to be a strict initial object (since the incoming morphism $h : Q \to X$ is not an isomorphism), and in the second case, Z ceases to be a strict terminal object (since the outgoing morphism $h : Z \to Q$ is also not an isomorphism), as demonstrated in Figure

 doi:10.20944/preprints202403.1479.v1

13. Indeed, X and Z are no longer initial/terminal objects even in the non-strict sense, due to the existence of multiple (currently algebraically inequivalent) outgoing morphisms:

$$(f : X \to Y) \neq \Big((f \circ h) \circ h^{-1} : X \to Y\Big), \qquad \text{and} \qquad (g \circ f : X \to Z) \neq \Big((g \circ f) \circ \Big(h \circ h^{-1}\Big) : X \to Z\Big), \quad (55)$$

from X to Y and from X to Z in the former case, and multiple (currently algebraically inequivalent) incoming morphisms:

$$(g : Y \to Z) \neq \Big(\Big(h^{-1} \circ h\Big) \circ g : Y \to Z\Big) \qquad \text{and} \qquad (g \circ f : X \to Z) \neq \Big(\Big(h^{-1} \circ h\Big) \circ (g \circ f) : X \to Z\Big), \quad (56)$$

from Y to Z and from X to Z in the latter case. However, if we now impose the pair of algebraic equivalences:

$$\Big(h \circ h^{-1} : X \to X\Big) = (id_X : X \to X), \qquad \text{and} \qquad \Big(h^{-1} \circ h : Q \to Q\Big) = (id_Q : Q \to Q), \quad (57)$$

thereby ensuring that:

$$\Big((f \circ h) \circ h^{-1} : X \to Y\Big) = (f : X \to Y), \qquad \text{and} \qquad \Big((g \circ f) \circ \Big(h \circ h^{-1}\Big) : X \to Z\Big) = (g \circ f : X \to Z), \quad (58)$$

in the former case, and the pair of algebraic equivalences:

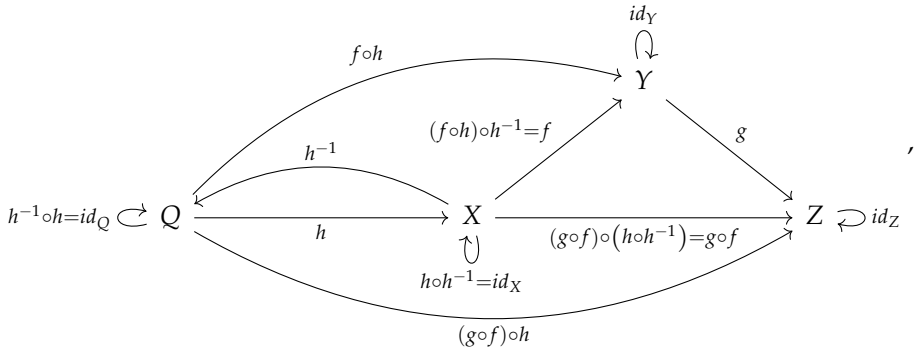
$$\Big(h^{-1} \circ h : Z \to Z\Big) = (id_Z : Z \to Z), \qquad \text{and} \qquad \Big(h \circ h^{-1} : Q \to Q\Big) = (id_Q : Q \to Q), \quad (59)$$

thereby ensuring that:

$$\Big(\Big(h^{-1} \circ h\Big) \circ g : Y \to Z\Big) = (g : Y \to Z), \qquad \text{and} \qquad \Big(\Big(h^{-1} \circ h\Big) \circ (g \circ f) : X \to Z\Big) = (g \circ f : X \to Z), \quad (60)$$

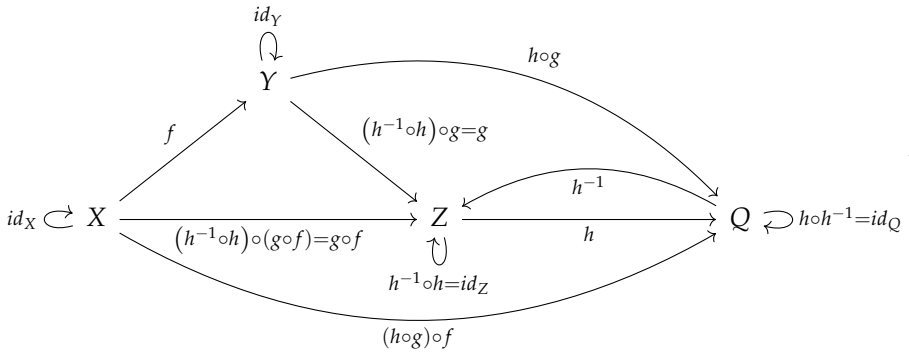
in the latter case, then morphisms $h : Q \to X$ and $h : Z \to Q$ (as well as their corresponding reverse morphisms $h^{-1} : X \to Q$ and $h^{-1} : Q \to Z$, respectively) now become isomorphisms, collapsing the above diagrams to:

 doi:10.20944/preprints202403.1479.v1



, (61)

and:



, (62)

respectively, and therefore causing X and Z to be strict initial and terminal objects, respectively, as illustrated in Figure 14. Any object that is both an initial object and a terminal object simultaneously is known as a *zero object* (and, by extension, any object that is both a strict initial object and a strict terminal object simultaneously is known as a *strict zero object*), and shares certain structural features in common with the point space object $*$ in the category $\mathbf{Top}_*$ of *pointed* topological spaces (i.e. topological spaces with a distinguished base point). Categories that contain zero objects are consequently referred to as *pointed categories*. CATEGORICA also has specialized functionality for handling zero objects, strict zero objects and pointed categories built into the `AbstractCategory` function.

 doi:10.20944/preprints202403.1479.v1

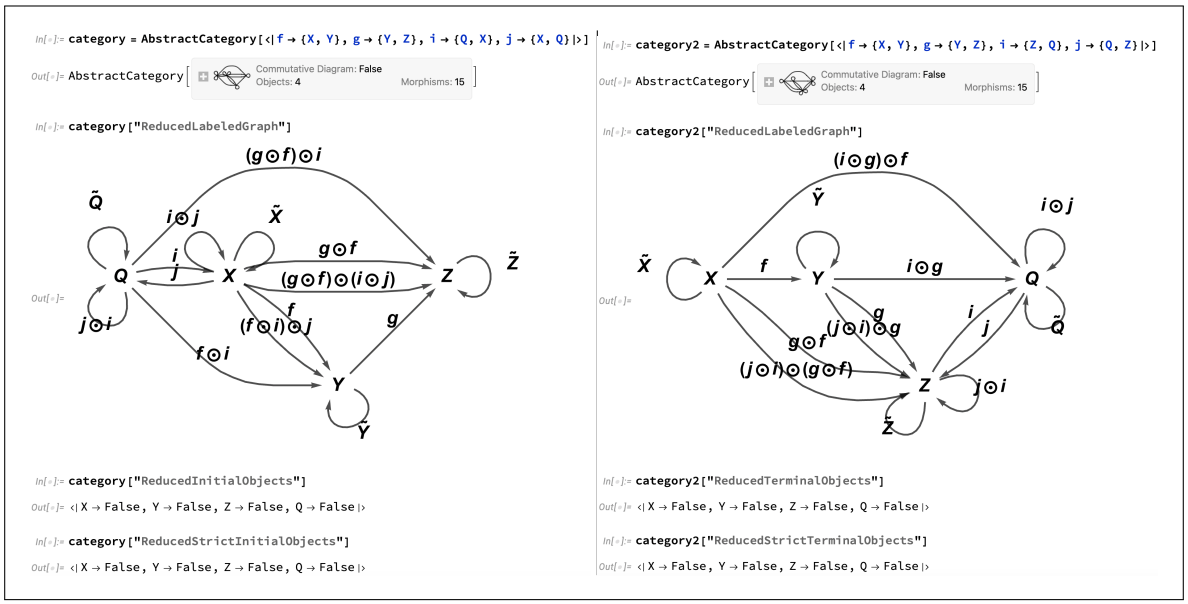


Figure 13. On the left, the `AbstractCategory` object corresponding to the simple triangular diagram from before, with new incoming and outgoing morphisms i and j added between the objects X and Q, showing that X has now ceased to be a (strict) initial object. On the right, the `AbstractCategory` object corresponding to the simple triangular diagram from before, with new incoming and outgoing morphisms i and j added between the objects Z and Q, showing that Z has now ceased to be a (strict) terminal object.

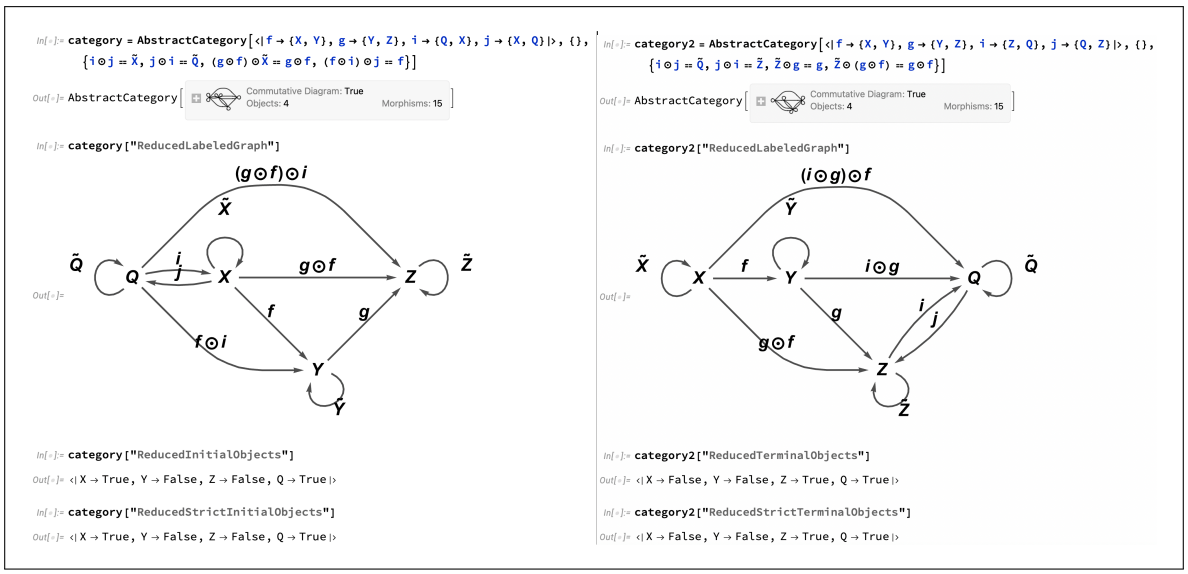


Figure 14. On the left, the `AbstractCategory` object corresponding to the simple triangular diagram from before, with new incoming and outgoing morphisms i and j added between the objects X and Q, along with the morphism equivalences $i \circ j = id_X$ and $j \circ i = id_Q$ (as well as their corollaries), showing that X is now a strict initial object. On the right, the `AbstractCategory` object corresponding to the simple triangular diagram from before, with incoming and outgoing morphisms i and j added between the objects Z and Q, along with the morphism equivalences $i \circ j = id_Q$ and $j \circ i = id_Z$ (as well as their corollaries), showing that Z is now a strict terminal object.

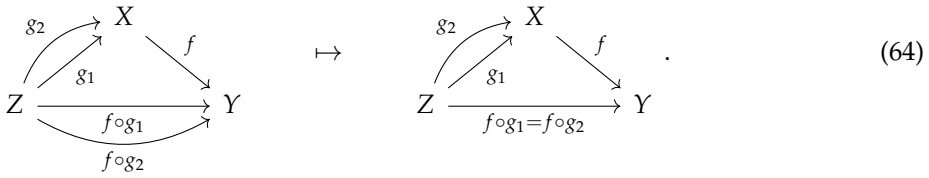
The final example of a dual construction that we shall discuss within the context of the present section, due to its close relationship with the monomorphism vs. epimorphism duality discussed

 doi:10.20944/preprints202403.1479.v1

previously, is that of *constant* vs. *coconstant* morphisms. If the morphism $f : X \to Y$ in the category $\mathcal{C}$ is such that, for all objects Z and all pairs of morphisms $g_1 : Z \to X$ and $g_2 : Z \to X$, one necessarily has that $(f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y)$, then $f : X \to Y$ is a constant morphism:

$$\begin{aligned} &\forall (f : X \to Y) \in \mathrm{hom}(\mathcal{C}), \qquad (f : X \to Y) \text{ is a constant morphism,} \\ &\quad \iff \qquad \forall Z \in \mathrm{ob}(\mathcal{C}), \qquad \forall (g_1 : Z \to X), (g_2 : Z \to X) \in \mathrm{hom}(\mathcal{C}), \\ &\qquad\qquad \text{one has} \qquad (f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y), \end{aligned} \tag{63}$$

or, illustrated diagrammatically, one has:

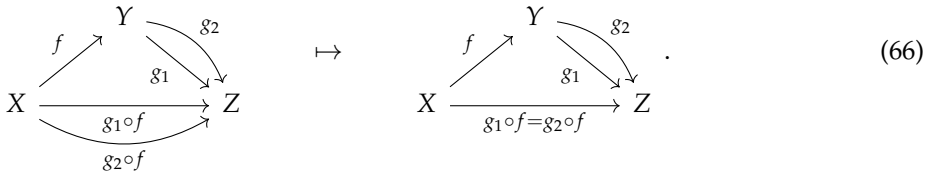


(64)

Note that this is identical to the diagram that appears in the *hypothesis* of the definition of a monomorphism (and therefore we may say, slightly loosely, that a monomorphism is a constant morphism for which one necessarily has that $(g_1 : Z \to X) = (g_2 : Z \to X)$). Dually, if the morphism $f : X \to Y$ in the category $\mathcal{C}$ is such that, for all objects Z and all pairs of morphisms $g_1 : Y \to Z$ and $g_2 : Y \to Z$, one necessarily has that $(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z)$, then $f : X \to Y$ is a coconstant morphism:

$$\begin{aligned} &\forall (f : X \to Y) \in \mathrm{hom}(\mathcal{C}), \qquad (f : X \to Y) \text{ is a coconstant morphism,} \\ &\quad \iff \qquad \forall Z \in \mathrm{ob}(\mathcal{C}), \qquad \forall (g_1 : Y \to Z), (g_2 : Y \to Z) \in \mathrm{hom}(\mathcal{C}), \\ &\qquad\qquad \text{one has} \qquad (g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z), \end{aligned} \tag{65}$$

or, illustrated diagrammatically, one has:



(66)

Note that this is, likewise, identical to the diagram that appears in the *hypothesis* of the definition of an epimorphism (and therefore we may also say, again slightly loosely, that an epimorphism is a coconstant morphism for which one necessarily has that $(g_1 : Y \to Z) = (g_2 : Y \to Z)$). Constant morphisms are so-named because they generalize the notion of constant functions in mathematical analysis (and, accordingly, coconstant morphisms may be thought of as generalizing the notion of zero-maps in analysis); any morphism that is both a constant morphism and a coconstant morphism is known as a *zero* morphism, since such morphisms share certain key algebraic properties with morphisms mapping into and out of zero objects (by the same token, constant morphisms share certain algebraic properties with morphisms mapping into terminal objects, and coconstant morphisms share certain algebraic properties with morphisms mapping out of initial objects) [47]. Figure 15 illustrates the basic diagrammatic setup for both constant and coconstant morphisms in CATEGORICA, and demonstrates that initially (in the absence of any further algebraic equivalences) the morphism $f : X \to Y$ is not a constant morphism (in the former case), or not a coconstant morphism (in the latter case), and hence also not a zero morphism (in either case). Figure 16 shows that, by imposing the algebraic equivalence $(f \circ g_1 : Z \to Y) = (f \circ g_2 : Z \to Y)$ in the former case and $(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z)$ in the

 doi:10.20944/preprints202403.1479.v1

latter case, one is able to force the morphism $f : X \to Y$ to be a constant morphism in the former example, and to be a coconstant morphism in the latter example, and therefore a zero morphism in both examples.

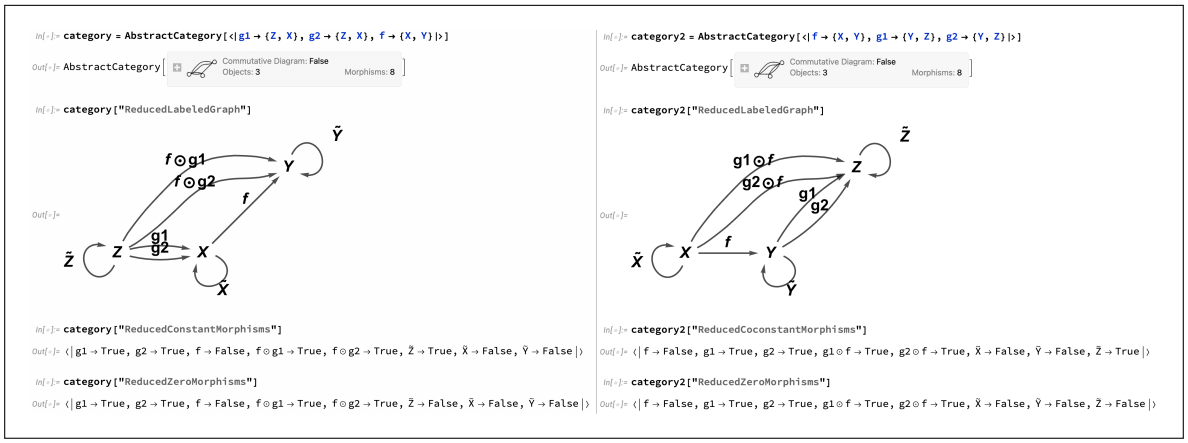


Figure 15. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a constant morphism, showing that initially morphism f is not a constant morphism, and hence also not a zero morphism. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a coconstant morphism, showing that initially morphism f is not a coconstant morphism, and hence also not a zero morphism.

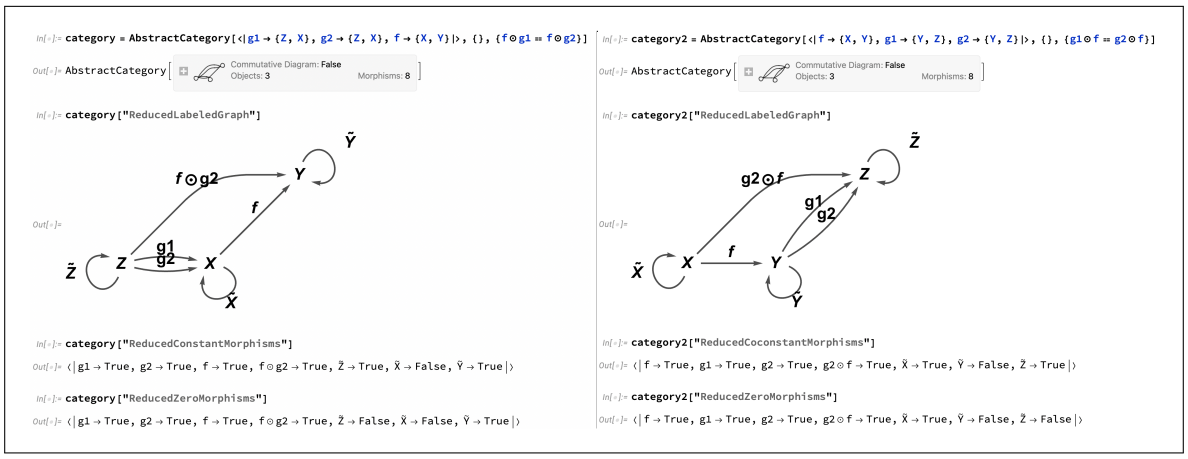


Figure 16. On the left, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a constant morphism, with the additional algebraic equivalence $f \circ g_1 = f \circ g_2$ imposed, showing that morphism f has now become a constant morphism, and hence also a zero morphism. On the right, the `AbstractCategory` object corresponding to the basic diagrammatic setup of a coconstant morphism, with the additional algebraic equivalence $g_1 \circ f = g_2 \circ f$ imposed, showing that morphism f has now become a coconstant morphism, and hence also a zero morphism.

The `AbstractCategory` function in CATEGORICA also contains functionality for detecting and manipulating many other special types of morphism, including *endomorphisms* (i.e. morphisms mapping an object X to itself) through the *"Endomorphisms"* property and its variants, *automorphisms* (i.e. isomorphisms between an object X and itself) through the *"Automorphisms"* property and its variants, etc. There are also many other special types of category (beyond simply the groupoids and commutative diagrams discussed thus far) that can be automatically detected and represented using CATEGORICA, with `AbstractCategory` in many cases including specialized functionality for manipulating them, including *discrete categories* (i.e. categories in which the only morphisms are identity morphisms)

 doi:10.20944/preprints202403.1479.v1

through the *"DiscreteCategoryQ"* property, *indiscrete categories* (i.e. categories in which there exists a unique morphism between every pair of objects) through the *"IndiscreteCategoryQ"* property, *balanced categories* (i.e. categories in which every bimorphism is also necessarily an isomorphism) through the *"BalancedCategoryQ"* property, etc. Needless to say, many additional properties and capabilities are also planned for future development. In many cases these properties have rather elegant mathematical interpretations; for instance, discrete categories may be thought as being a natural category-theoretic generalization of discrete topological spaces (i.e. topological spaces equipped with a *discrete topology*, in which every subset is open), since in a discrete topological space the only allowable paths are the identity paths from a point to itself, etc. However, in the interests of brevity, we will not cover these additional capabilities in any detail here.

4. Functors and Fibrations

A homomorphism (i.e. a structure-preserving map) between two categories is known as a *functor* [38,48] . More precisely, the map $F : \mathcal{C} \to \mathcal{D}$ from a category $\mathcal{C}$ to a category $\mathcal{D}$ is a functor if and only if it associates every object X in category $\mathcal{C}$ to a corresponding object $F(X)$ in category $\mathcal{D}$, i.e:

$$\forall X \in \mathrm{ob}(\mathcal{C}), \qquad \exists F(X) \in \mathrm{ob}(\mathcal{D}), \tag{67}$$

and every morphism $f : X \to Y$ in category $\mathcal{C}$ to a corresponding morphism $F(f) : F(X) \to F(Y)$ in category $\mathcal{D}$, i.e:

$$\forall (f : X \to Y) \in \mathrm{hom}(\mathcal{C}), \qquad \exists (F(f) : F(X) \to F(Y)) \in \mathrm{hom}(\mathcal{D}), \tag{68}$$

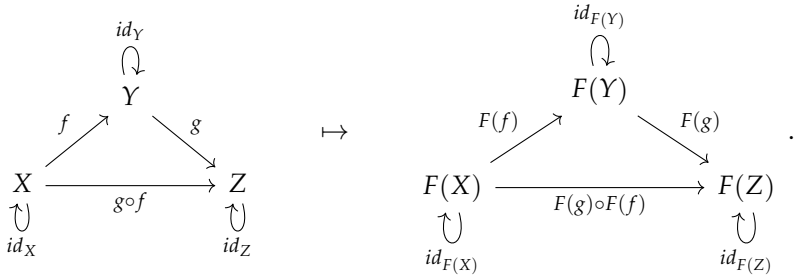
in such a way that the identity morphisms are all preserved, such that the identity morphism $id_X : X \to X$ on object X in category $\mathcal{C}$ is always mapped to the identity morphism $id_{F(X)} : F(X) \to F(X)$ on object $F(X)$ in category $\mathcal{D}$, i.e:

$$\forall X \in \mathrm{ob}(\mathcal{C}), \qquad (F(id_X) : F(X) \to F(X)) = \left(id_{F(X)} : F(X) \to F(X)\right), \tag{69}$$

and the composition of morphisms is also preserved, such that the composite morphism $g \circ f : X \to Z$ in category $\mathcal{C}$ (where $f : X \to Y$ and $g : Y \to Z$ are also morphisms in $\mathcal{C}$) is always mapped to the composite morphism $F(g) \circ F(f) : F(X) \to F(Z)$ in category $\mathcal{D}$, i.e:

$$\forall (f : X \to Y), (g : Y \to Z) \in \mathrm{hom}(\mathcal{C}), \\ (F(g \circ f) : F(X) \to F(Z)) = (F(g) \circ F(f) : F(X) \to F(Z)). \tag{70}$$

The functor construction may be illustrated diagrammatically by means of the following minimal example:



. (71)

Functoriality turns out to be an extremely powerful algebraic condition, and functors may consequently be thought of as generalizing many central constructions in pure mathematics, including group actions in the context of group theory (wherein one considers functors from single-object groupoids, i.e. groups, to the category **Set** whose objects are sets and whose morphisms are set-valued functions),

 doi:10.20944/preprints202403.1479.v1

group *representations* in the context of representation theory (wherein one again considers functors from groups/single-object groupoids, but now to the category **Vect** whose objects are vector spaces and whose morphisms are linear maps), and tangent bundles in differential geometry (wherein one considers functors from the category $\mathbf{Man}^p$ whose objects are differentiable manifolds of smoothness class C^p and whose morphisms are p-times continuously differentiable maps, to the category $\mathbf{Vect}(\mathbf{Man}^p)$ whose objects are vector bundles over the manifolds in $\mathbf{Man}^p$ and whose morphisms are vector bundle homomorphisms). Indeed, as mentioned previously, even categorical diagrams themselves may be formalized as functors $D : \mathcal{J} \to \mathcal{C}$ from some index category/scheme $\mathcal{J}$ to an arbitrary category $\mathcal{C}$. Power sets in set theory (i.e. functors from the category **Set** of sets and set-valued functions to itself) and fundamental groups in algebraic topology (i.e. functors from the category $\mathbf{Top}_*$ of pointed topological spaces and continuous functions between them, to the category **Grp** of groups and group homomorphisms) constitute further examples of key functorial constructions in mathematics that have previously been alluded to within this article. Figure 17 shows the implementation of the minimal example presented above, along with a slightly larger example, in the CATEGORICA framework using the `AbstractFunctor` function; much like with the `AbstractCategory` objects themselves, CATEGORICA automatically keeps track of, and enforces, all necessary algebraic equivalences between objects and morphisms within the codomain category of an `AbstractFunctor` object that must be imposed in order to maintain consistency with the functor axioms described above.

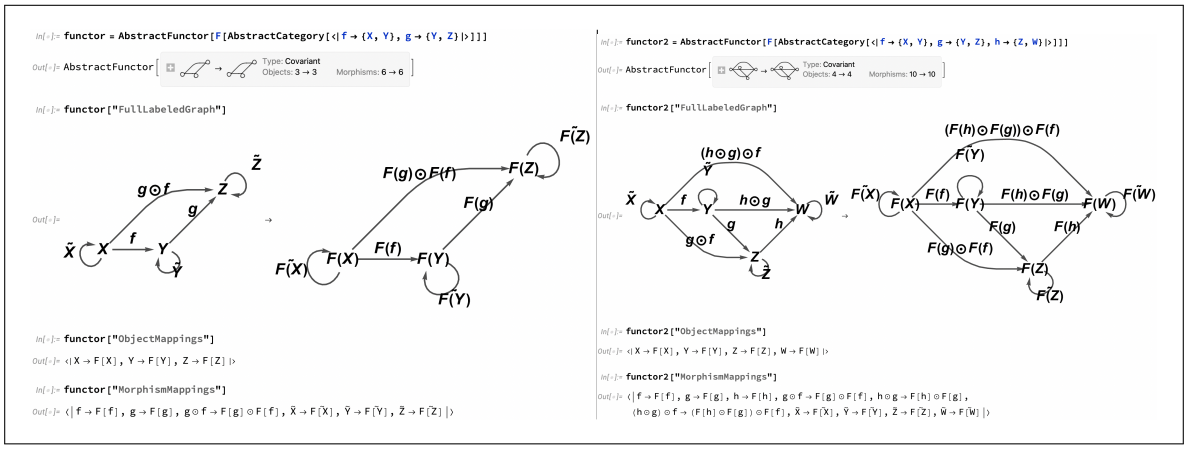


Figure 17. On the left, the `AbstractFunctor` object for a simple (three object, six-morphism) domain category, showing the explicit mappings between objects and morphisms in the domain and codomain categories. On the right, the `AbstractFunctor` object for a slightly larger (four-object, ten-morphism) domain category, showing the explicit mappings between objects and morphisms in the domain and codomain categories.

Although the formal definition presented above corresponds to the case of *covariant* functors (namely functors which preserve the directions of morphisms, and hence which also preserve the order of composition of morphisms), there also exists a dual notion of *contravariant* functors, which obey analogous axioms but which reverse the directions of morphisms, and hence also reverse the order of morphism composition. The definition of the mapping on objects remains unchanged from the covariant case, but now every morphism $f : X \to Y$ in category $\mathcal{C}$ is mapped to a corresponding morphism $F(f) : F(Y) \to F(X)$ in category $\mathcal{D}$, i.e:

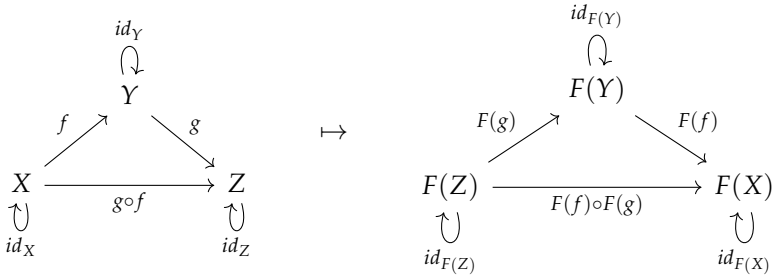
$$\forall (f : X \to Y) \in \hom(\mathcal{C}), \qquad \exists (F(f) : F(Y) \to F(X)) \in \hom(\mathcal{D}). \tag{72}$$

The condition on identity morphisms is also unchanged from the covariant case, but now the composite morphism $g \circ f : X \to Z$ in category $\mathcal{C}$ (where $f : X \to Y$ and $g : Y \to Z$ are also morphisms in $\mathcal{C}$) is always mapped to the composite morphism $F(f) \circ F(g) : F(Z) \to F(X)$ in category $\mathcal{D}$, i.e:

 doi:10.20944/preprints202403.1479.v1

$$\forall (f : X \to Y), (g : Y \to Z) \in \hom(\mathcal{C}),$$
$$(F(g \circ f) : F(Z) \to F(X)) = (F(f) \circ F(g) : F(Z) \to F(X)). \quad (73)$$

The contravariant functor construction may, just as before, be illustrated diagrammatically by means of the following minimal example:



(74)

Just as covariant functors from the category $\mathbf{Man}^p$ of differentiable manifolds and differentiable maps to the category $\mathbf{Vect}(\mathbf{Man}^p)$ of vector bundles and vector bundle homomorphisms may be used to generalize the construction of tangent bundles in differential geometry, the construction of *cotangent* bundles may correspondingly be generalized using the corresponding *contravariant* functor from $\mathbf{Man}^p$ (or, more precisely, from its dual/opposite category $(\mathbf{Man}^p)^{op}$) to $\mathbf{Vect}(\mathbf{Man}^p)$. Likewise, the operation of taking dual spaces in linear algebra corresponds to the application of a certain contravariant functor from the category $\mathbf{Vect}(K)$ of vector spaces and linear maps over some fixed field K (or, more precisely, from its dual/opposite category $(\mathbf{Vect}(K))^{op}$), to itself. Indeed, the power set construction in set theory may in fact be formalized as *either* a covariant functor $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ (as above), *or* as a contravariant functor $\mathcal{P} : \mathbf{Set}^{op} \to \mathbf{Set}$. The notion of presheaves on a space X in algebraic topology may be formalized as a functor from the dual/opposite category $\mathcal{C}^{op}$ of the category $\mathcal{C}$ whose objects are open sets $U \subseteq X$ and whose morphisms are inclusion maps $U \subseteq V$, to $\mathbf{Set}$; when appropriately extended to the case where the category $\mathcal{C}$ (and hence the domain category $\mathcal{C}^{op}$ of the contravariant functor) is arbitrary, this yields the corresponding category-theoretic notion of a presheaf, which is far more general. Figure 18 demonstrates a CATEGORICA implementation of the corresponding contravariant versions of the two functors previously shown in Figure 17, obtained in each case by setting the first argument of `AbstractFunctor` (i.e. the covariance argument) to `False`. At any time, the dual version of any `AbstractFunctor` object may be computed directly using the *"SwapVariance"* property (analogous to the *"DualCategory"* property of `AbstractCategory` objects), with CATEGORICA automatically making any required modifications to the order of morphism composition within all morphism equivalence lists, etc.

 doi:10.20944/preprints202403.1479.v1

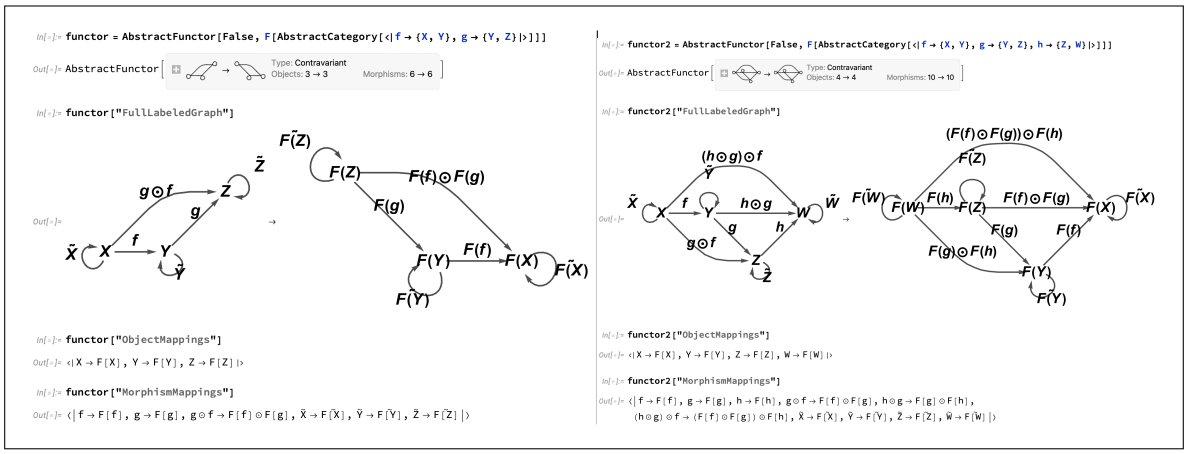


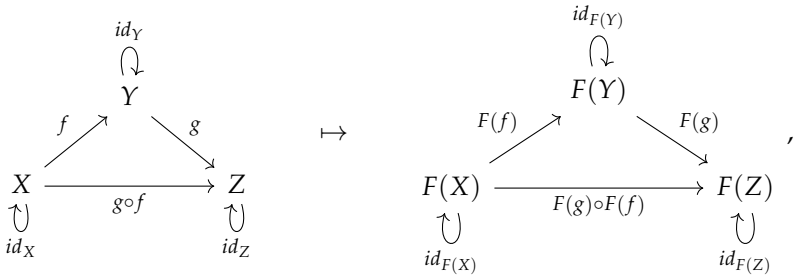
Figure 18. On the left, the contravariant `AbstractFunctor` object for a simple (three-object, six-morphism) domain category, showing the explicit mappings between objects and morphisms in the (dual) domain and codomain categories, illustrating reversal of the order of morphism composition. On the left, the contravariant `AbstractFunctor` object for a slightly larger (four-object, ten-morphism) domain category, showing the explicit mappings between objects and morphisms in the (dual) domain and codomain categories, illustrating reversal of the order of morphism composition.

In its full form, and in the absence of any additional variance information, an `AbstractFunctor` object is specified by the `AbstractCategory` object for its domain category, an association of object mappings, an association of morphism (or, more precisely, arrow) mappings, a list of new objects in the codomain category, a list of new morphisms (or, more precisely, arrows) in the codomain category, a list of new object equivalences in the codomain category, and a list of new morphism equivalences in the codomain category (as well as optional additional information regarding choices of composition and identity symbols, as with `AbstractCategory`). Note, in particular, that all mappings take place at the level of the underlying `AbstractQuiver` object, rather than on the `AbstractCategory` object itself, so as to guarantee consistency with the composition/functoriality axioms. All algebraic equivalences between objects and arrows/morphisms in the domain `AbstractCategory` object (or its underlying `AbstractQuiver` object) are automatically translated into corresponding algebraic equivalences in the codomain `AbstractCategory` object. We may therefore regard a functor $F : \mathcal{C} \to \mathcal{D}$ as corresponding to a pair of (set-valued, since we are dealing here with small categories) functions, F_{ob} and F_{hom}, with the former mapping the set of objects in category $\mathcal{C}$ to the set of objects in category $\mathcal{D}$ and the latter mapping the set of morphisms in category $\mathcal{C}$ to the set of morphisms in category $\mathcal{D}$:

$$F_{ob} : \mathrm{ob}(\mathcal{C}) \to \mathrm{ob}(\mathcal{D}), \qquad \text{and} \qquad F_{hom} : \mathrm{hom}(\mathcal{C}) \to \mathrm{hom}(\mathcal{D}). \tag{75}$$

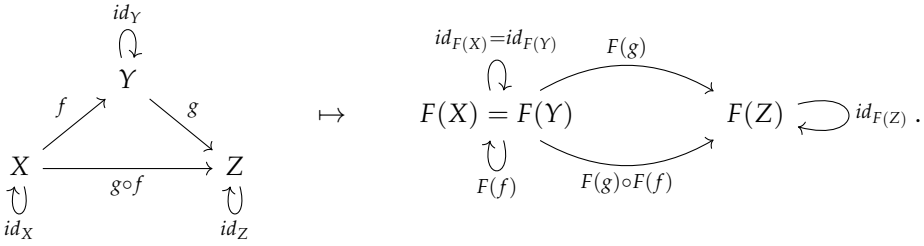
This, in turn, presents (at least) two distinct ways in which functors may be classified as either injective, surjective or bijective; namely, functors may be injective/surjective/bijective on objects, or on morphisms, or on both, or on neither (depending upon the injectivity/surjectivity/bijectivity properties of the functions F_{ob} and F_{hom}). A simple way in which a functor may fail to be injective on objects is by having two objects in the codomain category be equivalent which were not previously equivalent in the domain category; for instance, consider the functor:

 doi:10.20944/preprints202403.1479.v1



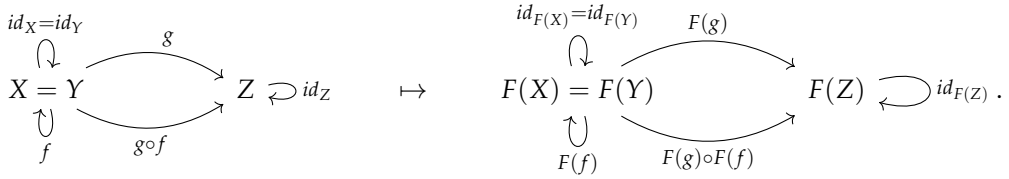
(76)

plus the additional algebraic condition that $F(X) = F(Y)$, imposed on the objects in the codomain category $\mathcal{D}$, such that one instead has:



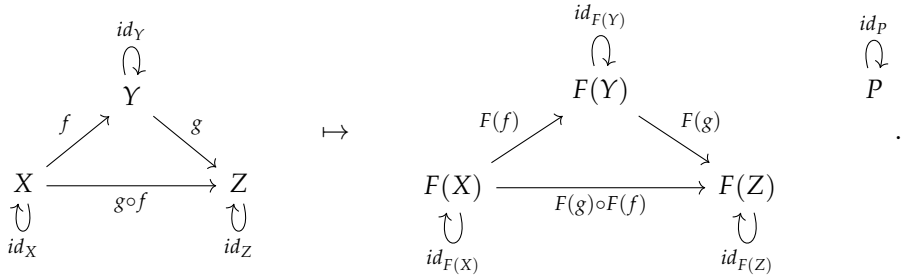
(77)

Such a functor would no longer be injective, and hence also no longer bijective, on objects, although its injectivity (and therefore bijectivity) on objects could nevertheless be restored by introducing a corresponding algebraic condition $X = Y$ on the objects in the domain category $\mathcal{C}$, such that one instead has:



(78)

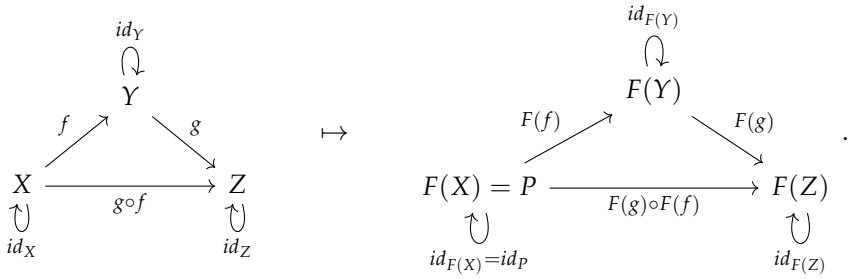
Figure 19 implements this basic example in CATEGORICA, demonstrating how functor injectivity on objects may be removed (by imposing the algebraic equivalence $F(X) = F(Y)$ on objects in the codomain category) and then subsequently reinstated (by imposing the additional algebraic equivalence $X = Y$ on objects in the domain category). Dual to this construction, a simple way in which a functor may fail to be surjective on objects is by having a new object be introduced in the codomain category that did not previously exist in the domain category; for instance, consider now the functor:



(79)

Such a functor would no longer be surjective, and hence also no longer bijective, on objects, although its surjectivity (and therefore bijectivity) on objects could nevertheless be restored by introducing a new algebraic condition $P = F(X)$ on the objects in the codomain category $\mathcal{D}$, such that one instead has:

 doi:10.20944/preprints202403.1479.v1



(80)

Figure 20 implements this dual example in CATEGORICA, demonstrating now how functor surjectivity on objects may be removed (by introducing the new object P in the codomain category) and then subsequently reinstated (by imposing the algebraic equivalence $P = F(X)$ on objects in the codomain category), in much the same way.

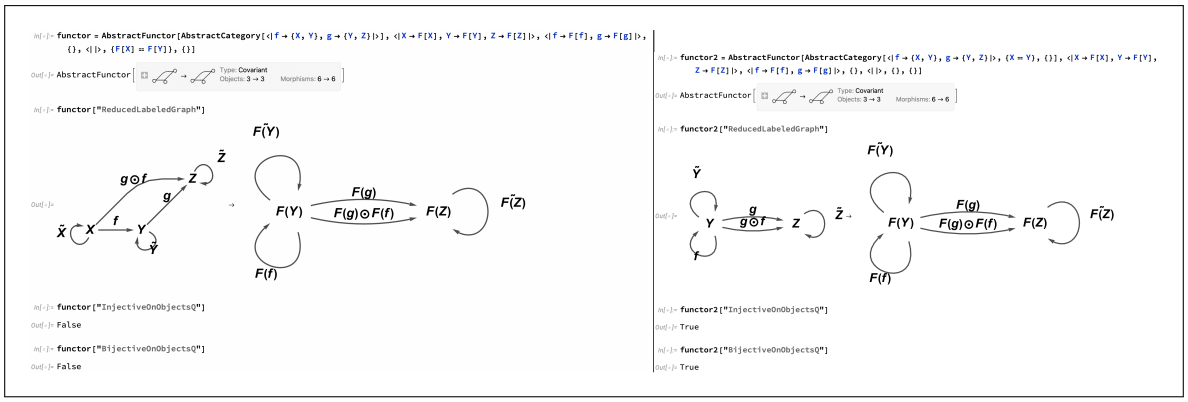


Figure 19. On the left, an `AbstractFunctor` object which is not injective, and hence not bijective, on objects because of the additional algebraic equivalence $F(X) = F(Y)$ being imposed on the objects of the codomain category. On the right, the corresponding `AbstractFunctor` object with injectivity, and hence bijectivity, on objects restored, by imposing the additional algebraic equivalence $X = Y$ on the objects of the domain category.

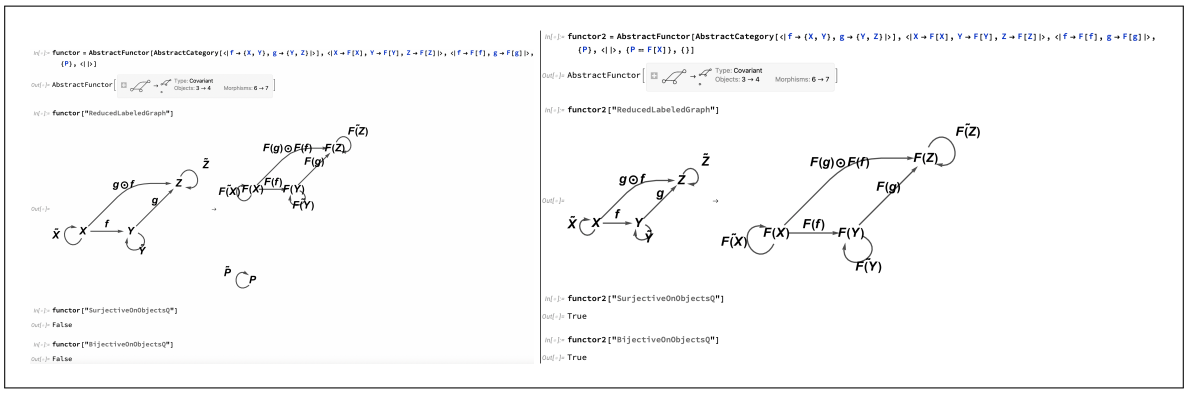
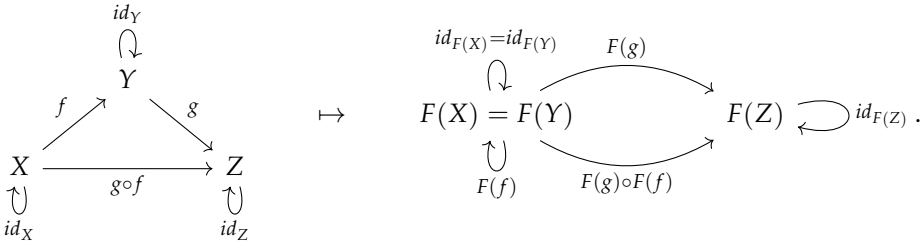


Figure 20. On the left, an `AbstractFunctor` object which is not surjective, and hence not bijective, on objects because of the additional object P being introduced within the codomain category. On the right, the corresponding `AbstractFunctor` object with surjectivity, and hence bijectivity, on objects restored, by imposing the additional algebraic equivalence $P = F(X)$ on the objects of the codomain category.

However, there may nevertheless be cases in which a functor is only injective, surjective or bijective on objects *up to isomorphism*, which in turn motivates the concepts of *essential injectivity*, *essential surjectivity* and *essential bijectivity*. For instance, consider again the example presented above, in

 doi:10.20944/preprints202403.1479.v1

which a functor fails to be injective on objects because of an additional algebraic condition $F(X) = F(Y)$ that has been imposed on the objects in the codomain category $\mathcal{D}$:

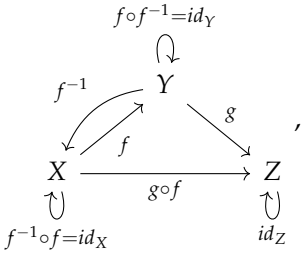


(81)

Now, rather than imposing the strict algebraic equality $X = Y$ on the objects in the domain category $\mathcal{C}$, suppose instead that one introduces a new inverse morphism $f^{-1} : Y \to X$ in $\mathcal{C}$, such that:

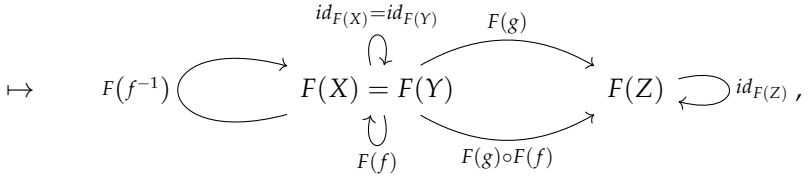
$$\left(f \circ f^{-1} : Y \to Y\right) = (id_Y : Y \to Y), \qquad \text{and} \qquad \left(f^{-1} \circ f : X \to X\right) = (id_X : X \to X), \tag{82}$$

and therefore the objects X and Y become isomorphic, thus yielding:



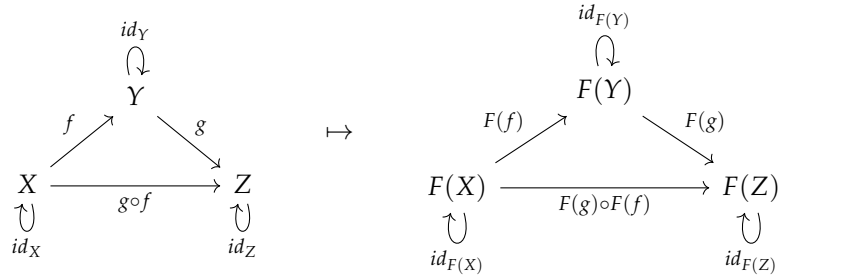
(83)

in the domain category, which maps to:



(84)

in the codomain category. Then, the resulting functor would still not be strictly injective, nor strictly bijective, on objects, but it would be *essentially* injective, and hence *essentially* bijective (i.e. injective/bijective up to isomorphism). Figure 21 implements this example in CATEGORICA, showing how essential injectivity may be reinstated by imposing the isomorphism $X \cong Y$ on objects in the domain category. Dually, consider again the previous example of a functor that fails to be surjective on objects because of a new object P that has been introduced in the codomain category:



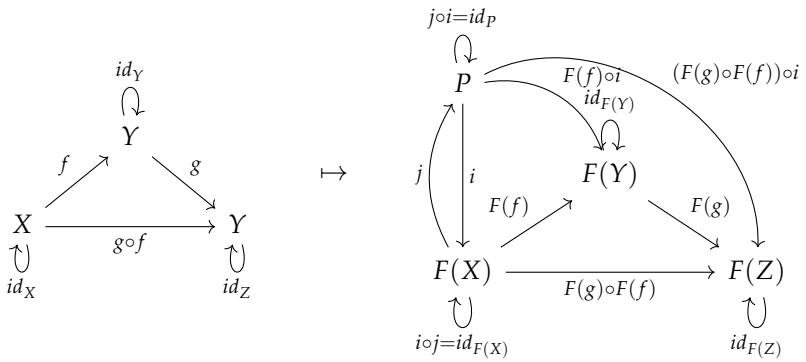
(85)

Similarly, rather than imposing the strict algebraic equality $P = F(X)$ on the objects in the codomain category $\mathcal{D}$, we can instead introduce a new pair of morphisms $i : P \to F(X)$ and $j : F(X) \to P$, such that:

 doi:10.20944/preprints202403.1479.v1

$$(i \circ j : F(X) \to F(X)) = \Big(id_{F(X)} : F(X) \to F(X)\Big), \qquad \text{and} \qquad (j \circ i : P \to P) = (id_P : P \to P), \quad (86)$$

and therefore force the objects P and $F(X)$ to be isomorphic, hence yielding:



(87)

Now, the resulting functor is still not strictly surjective, nor strictly bijective, on objects, but it is *essentially* surjective, and hence *essentially* bijective (i.e. surjective/bijective up to isomorphism). Figure 22 implements this dual example in CATEGORICA, showing how essential surjectivity may be reinstated by imposing the isomorphism $P \cong F(X)$ on objects in the codomain category.

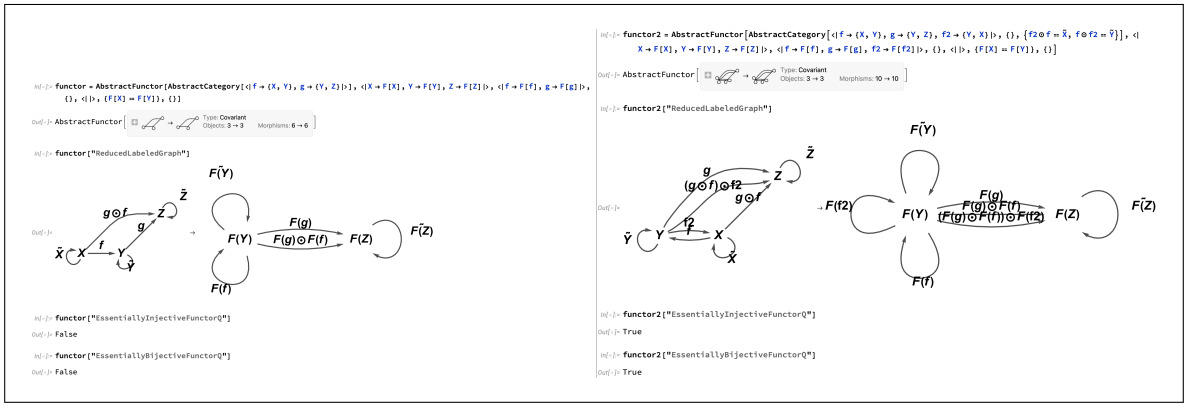


Figure 21. On the left, an `AbstractFunctor` object which is not essentially injective, and hence not essentially bijective, because of the additional algebraic equivalence $F(X) = F(Y)$ being imposed on the objects of the codomain category. On the right, the corresponding `AbstractFunctor` object with essential injectivity, and hence essential bijectivity, restored, by introducing a new isomorphism $X \cong Y$ between objects of the domain category.

 doi:10.20944/preprints202403.1479.v1

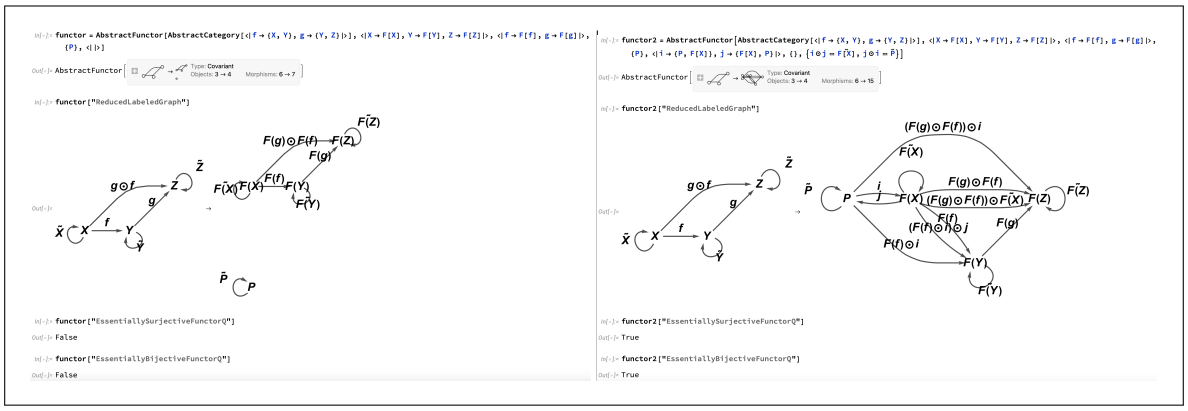
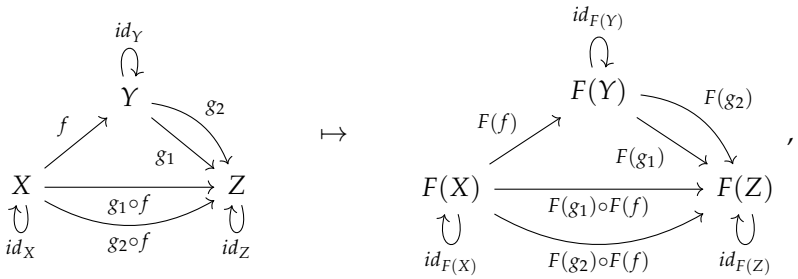


Figure 22. On the left, an `AbstractFunctor` object which is not essentially surjective, and hence not essentially bijective, because of the additional object P being introduced within the codomain category. On the right, the corresponding `AbstractFunctor` object with essential sujrectivity, and hence essential bijectivity, restored, by introducing a new isomorphism $P \cong F(X)$ between objects of the codomain category.

On the other hand, shifting now from considering the properties of the F_{ob} function to those of the F_{hom} function, functors that are injective, surjective or bijective on *morphisms* are known as *faithful* functors, *full* functors or *fully faithful* functors, respectively. A simple way in which a functor may fail to be faithful (i.e. fail to be injective on morphisms) is by having two morphisms in the codomain category be equivalent which were not previously equivalent in the domain category; for instance, consider the functor:

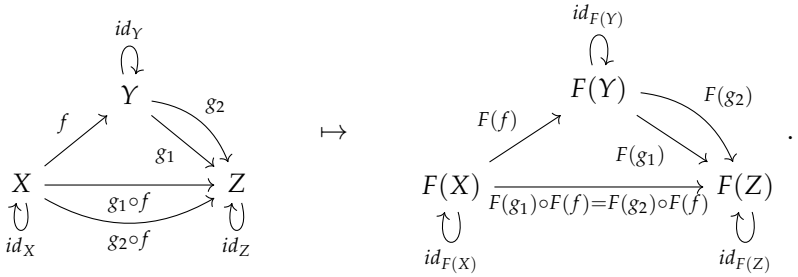


(88)

plus the additional algebraic condition that:

$$(F(g_1) \circ F(f) : F(X) \to F(Z)) = (F(g_2) \circ F(f) : F(X) \to F(Z)), \tag{89}$$

on the morphisms in the codomain category $\mathcal{D}$, such that one instead has:



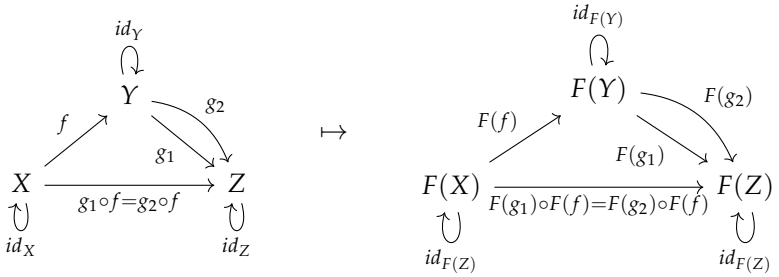
(90)

Such a functor would no longer be faithful (i.e. no longer injective on morphisms), and hence also no longer fully faithful (i.e. no longer bijective on morphisms), although its faithfulness, and therefore full faithfulness, could nevertheless be restored by introducing a corresponding algebraic condition:

 doi:10.20944/preprints202403.1479.v1

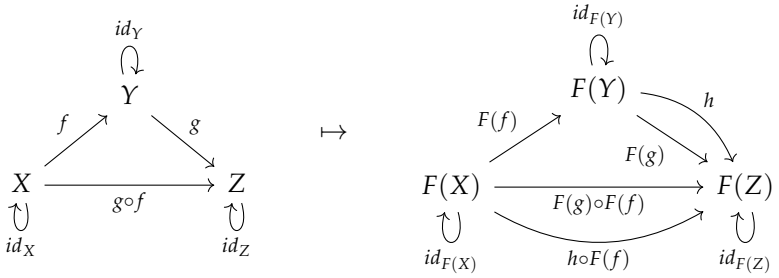
$$(g_1 \circ f : X \to Z) = (g_2 \circ f : X \to Z), \tag{91}$$

on the morphisms in the domain category $\mathcal{C}$, such that one now has:



(92)

Figure 23 implements this elementary example in CATEGORICA, illustrating how the faithfulness of a functor may be removed (by imposing the algebraic equivalence $F(g_1) \circ F(f) = F(g_2) \circ F(f)$ on morphisms in the codomain category) and then subsequently reinstated (by imposing the additional algebraic equivalence $g_1 \circ f = g_2 \circ f$ on morphisms in the domain category). Dual to this construction, a simple way in which a functor may fail to be full (i.e. fail to be surjective on morphisms) is by having a new morphism be introduced in the codomain category that did not previously exist in the domain category; for instance, consider now the functor:

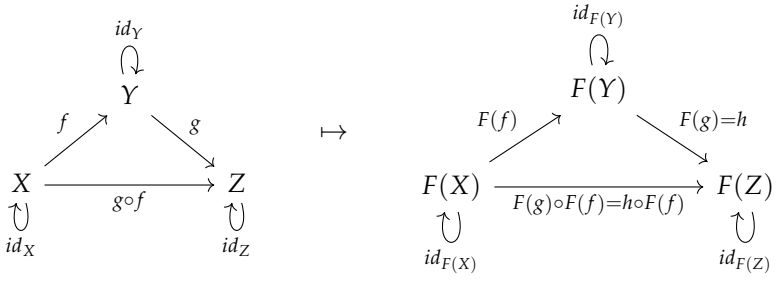


(93)

Such a functor would no longer be full (i.e. no longer surjective on morphisms), and hence also no longer fully faithful (i.e. no longer bijective on morphisms), although its fullness, and hence full faithfulness, could nevertheless be restored by introducing a new algebraic condition:

$$(F(g) : F(Y) \to F(Z)) = (h : F(Y) \to F(Z)), \tag{94}$$

on morphisms in the codomain category $\mathcal{D}$, such that one now has:



(95)

Figure 24 implements this dual example in CATEGORICA, illustrating how the fullness of a functor may be removed (by introducing the new morphism $h : F(Y) \to F(Z)$ in the codomain category) and then subsequently reinstated (by imposing the algebraic equivalence $F(g) = h$ on morphisms in the codomain category).

 doi:10.20944/preprints202403.1479.v1

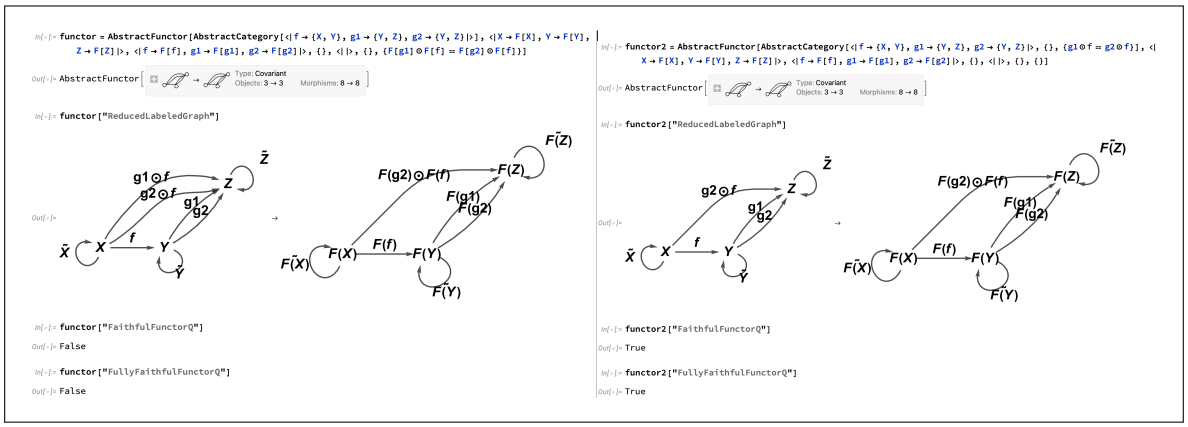


Figure 23. On the left, an `AbstractFunctor` object which is not faithful, and hence not fully faithful, because of the additional algebraic equivalence $F(g_1) \circ F(f) = F(g_2) \circ F(f)$ being imposed on the morphisms of the codomain category. On the right, the corresponding `AbstractFunctor` object with faithfulness, and hence full faithfulness, restored, by imposing the additional algebraic equivalence $g_1 \circ f = g_2 \circ f$ on the morphisms of the domain category.

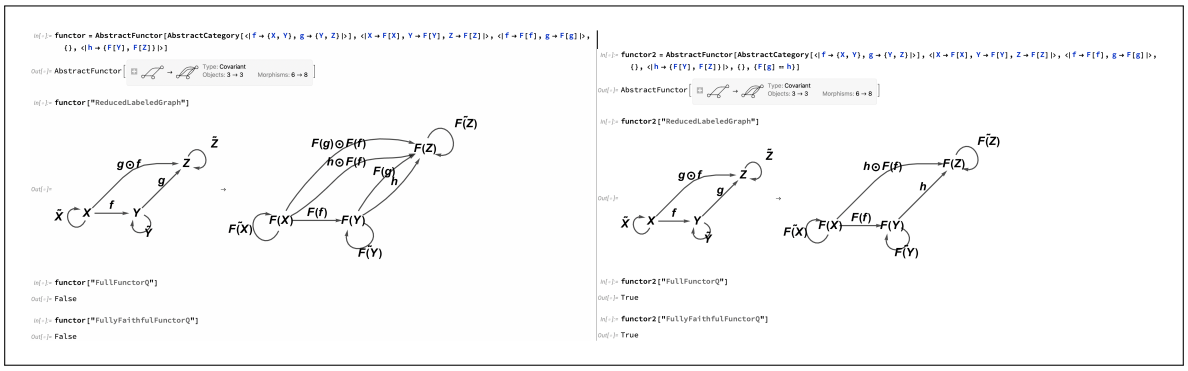


Figure 24. On the left, an `AbstractFunctor` object which is not full, and hence not fully faithful, because of the additional morphism h being introduced within the codomain category. On the right, the corresponding `AbstractFunctor` object with fullness, and hence full faithfulness, restored, by imposing the additional algebraic equivalence $F(g) = h$ on the morphisms of the codomain category.

The final example of a functorial construction that we shall cover within this section, due to its rather foundational significance in the fields of algebraic geometry, algebraic topology and (higher) homotopy theory, is that of a (Grothendieck) *fibration* [49,50]. This purely category-theoretic notion of a fibration is effectively a grand generalization of the notion of a fiber bundle (or topological fibration) in topology, wherein one considers a functor of the general form $F : \mathcal{E} \to \mathcal{B}$, with the domain category $\mathcal{E}$ being the *total category* of the fibration (thus playing an analogous role to that of the *total space* of a fiber bundle) and the codomain category $\mathcal{B}$ being the *base category* (thus playing an analogous role to that of the *base space* of the same fiber bundle). For each object X in the base category $\mathcal{B}$, one can consider the *fiber category* F_X at X, which is a category whose object set $\mathrm{ob}(F_X)$ is the set of objects in the total category $\mathcal{E}$ that map to X under the functor F (i.e. the preimage of X under the function F_{ob}) and whose morphism set $\hom(F_X)$ is the set of morphisms in the total category $\mathcal{E}$ that map to the identity morphism $id_X : X \to X$ on X under F (i.e. the preimage of $id_X : X \to X$ under the function F_{hom}), i.e:

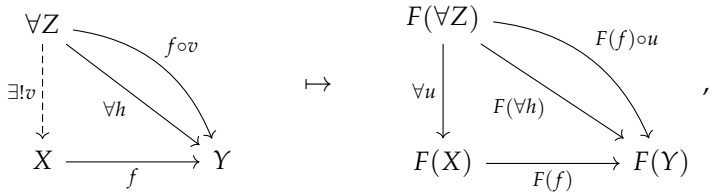
 doi:10.20944/preprints202403.1479.v1

$$\begin{aligned}\forall X \in \mathrm{ob}(\mathcal{B}), \qquad & F_X \text{ is the fiber category of functor } F : \mathcal{E} \to \mathcal{B} \text{ at } X \\ & \iff \mathrm{ob}(F_X) = \{U \in \mathrm{ob}(\mathcal{E}) : F(U) = X\}, \qquad \text{and} \\ & \mathrm{hom}(F_X) = \{(f : U \to V) \in \mathrm{hom}(\mathcal{E}) : (F(f) : F(U) \to F(V)) = (id_X : X \to X)\}. \end{aligned} \tag{96}$$

In order to qualify as a valid Grothendieck fibration, however, the functor $F : \mathcal{E} \to \mathcal{B}$ must satisfy an additional, and rather technical, condition known as *contravariant pseudofunctoriality* (for the corresponding dual construction, known as a Grothendieck *opfibration*, the relevant condition is *covariant pseudofunctoriality*), which is typically expressed in terms of *Cartesian morphisms*. The formal definition of a Cartesian morphism is unfortunately a little opaque; specifically, if the morphism $f : X \to Y$ in the total category $\mathcal{E}$ is such that, for all objects Z and all morphisms $h : Z \to Y$ in $\mathcal{E}$, and all morphisms $u : F(Z) \to F(X)$ in the base category $\mathcal{B}$ such that $(F(f) \circ u : F(Z) \to F(Y)) = (F(h) : F(Z) \to F(Y))$, there necessarily exists a unique morphism $v : Z \to X$ in the total category $\mathcal{E}$ such that $(f \circ v : Z \to Y) = (h : Z \to Y)$ in $\mathcal{E}$ and $(u : F(Z) \to F(X)) = (F(v) : F(X) \to F(Y))$ in $\mathcal{B}$, then $f : X \to Y$ is Cartesian:

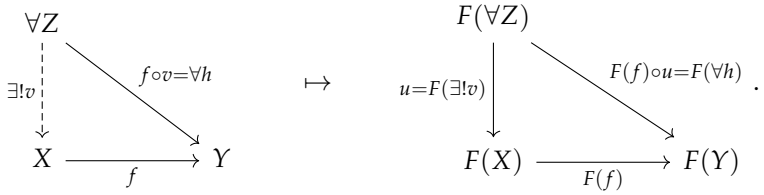
$$\begin{aligned}&\forall (f : X \to Y) \in \mathrm{hom}(\mathcal{E}), \qquad (f : X \to Y) \text{ is a Cartesian morphism} \\ &\quad \iff \qquad \forall Z \in \mathrm{ob}(\mathcal{E}), \qquad \forall (h : Z \to Y) \in \mathrm{hom}(\mathcal{E}), \qquad \forall (u : F(Z) \to F(X)) \in \mathrm{hom}(\mathcal{B}), \\ &(F(f) \circ u : F(Z) \to F(Y)) = (F(h) : F(Z) \to F(Y)) \qquad \implies \qquad \exists ! (v : Z \to X) \in \mathrm{hom}(\mathcal{E}), \\ &\qquad \text{such that} \qquad (f \circ v : Z \to Y) = (h : Z \to Y), \qquad \text{and} \\ &\qquad\qquad (u : F(Z) \to F(X)) = (F(v) : F(Z) \to F(X)), \end{aligned} \tag{97}$$

or, illustrated diagrammatically, one has the following basic setup:



, (98)

which then collapses down to:



. (99)

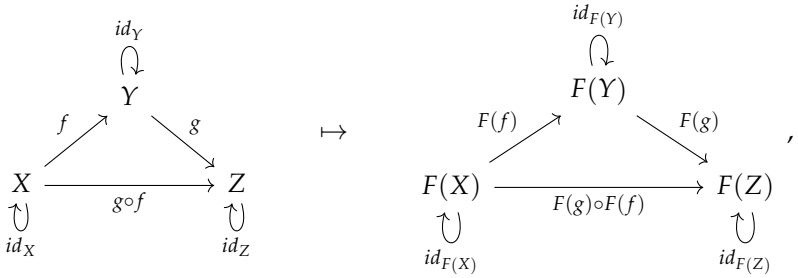
The contravariant pseudofunctoriality condition that characterizes Grothendieck fibrations is then the condition that, for every object Y in the total category $\mathcal{E}$, and every morphism $f_0 : X_0 \to F(Y)$ mapping to the image $F(Y)$ of Y in the base category $\mathcal{B}$, there must exist some Cartesian morphism $f : X \to Y$ in the total category $\mathcal{E}$ such that the morphism $f_0 : X_0 \to F(Y)$ is the image of the Cartesian morphism $f : X \to Y$ in the base category $\mathcal{B}$, i.e. $(F(f) : F(X) \to F(Y)) = (f_0 : X_0 \to F(Y))$ (and therefore also $F(X) = X_0$):

$$\begin{aligned}&\forall Y \in \mathrm{ob}(\mathcal{E}), \qquad \forall X_0 \in \mathrm{ob}(\mathcal{B}), \qquad \forall (f_0 : X_0 \to F(Y)) \in \mathrm{hom}(\mathcal{B}), \\ &\quad \exists X \in \mathrm{ob}(\mathcal{E}), \qquad \exists (f : X \to Y), \qquad \text{such that} \qquad (f : X \to Y) \text{ is Cartesian}, \qquad \text{and} \\ &\qquad\qquad F(X) = X_0, \qquad \text{and} \qquad (F(f) : X_0 \to F(Y)) = (f_0 : X_0 \to F(Y)). \end{aligned} \tag{100}$$

 doi:10.20944/preprints202403.1479.v1

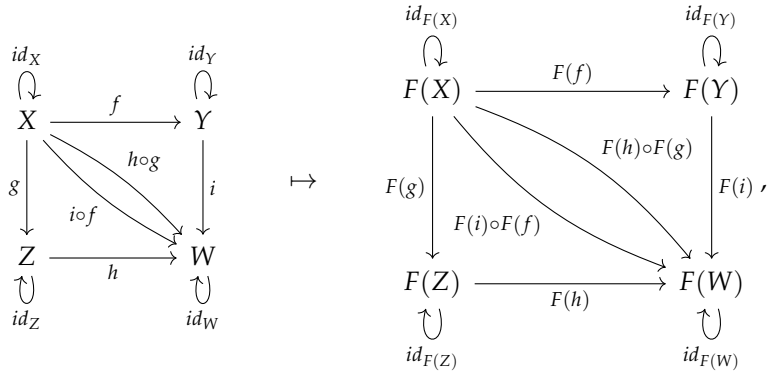
There exist many notable specializations of this highly abstract Grothendieck fibration definition, including the case of *discrete fibrations* (in which every fiber category F_X is a discrete category, consisting solely of objects and their identity morphisms) and *groupoidal fibrations* (in which every fiber category F_X is a groupoid). CATEGORICA does not yet possess the functionality to detect and characterize all cases of Grothendieck fibrations with complete generality, although important special cases (such as discrete fibrations, via the *"DiscreteFibrationQ"* property, etc.) have indeed been implemented fully.

For instance, if we consider the following simple functor from a three-object, six-morphism *total category* $\mathcal{E}$ to another three-object, six-morphism *base category* $\mathcal{B}$:



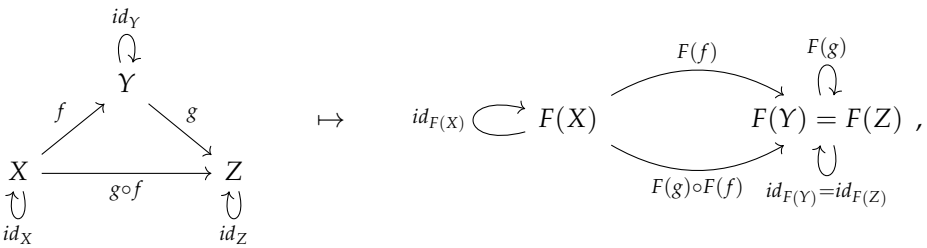
(101)

or, alternatively, the following slightly more complex functor from a four-object, ten-morphism *total category* $\mathcal{E}$ to another four-object, ten-morphism *base category* $\mathcal{B}$:



(102)

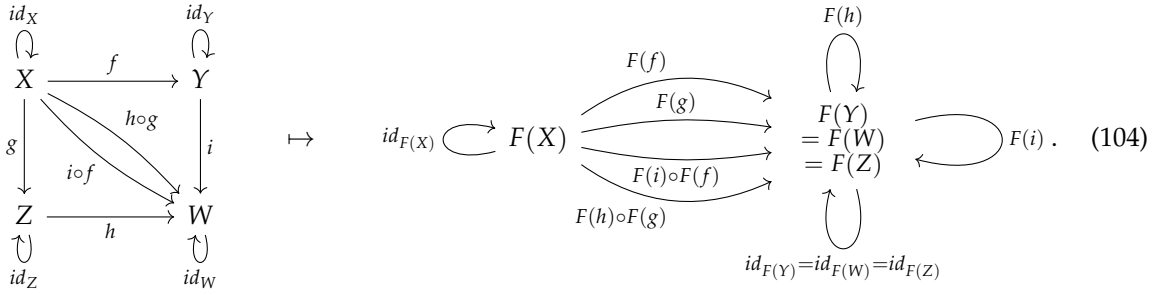
then we see that these functors both constitute more-or-less trivial cases of (discrete) fibrations. In the former case, there are three (discrete) fiber categories, one for each object $F(X)$, $F(Y)$ and $F(Z)$ in the base category $\mathcal{B}$, and each consisting of the single object X, Y or Z from the total category $\mathcal{E}$ (along with the corresponding identity morphism in each case), respectively; in the latter case, there are instead four (discrete) fiber categories, one for each object $F(X)$, $F(Y)$, $F(Z)$ and $F(W)$ in the base category $\mathcal{B}$, and each consisting of the single object X, Y, Z or W from the total category $\mathcal{E}$ (along with the corresponding identity morphism in each case), respectively. This is illustrated in Figure 25, in which CATEGORICA correctly identifies that the relevant `AbstractFunctor` objects are both discrete fibrations, and proceeds to compute the relevant fiber categories (represented as an association of `AbstractCategory` objects - one for each object in the codomain/base category of the fibration). In the first instance, imposing the additional algebraic equivalence $F(Y) = F(Z)$ on objects in the codomain category now yields a fibration over a two-object, five-morphism base category:



(103)

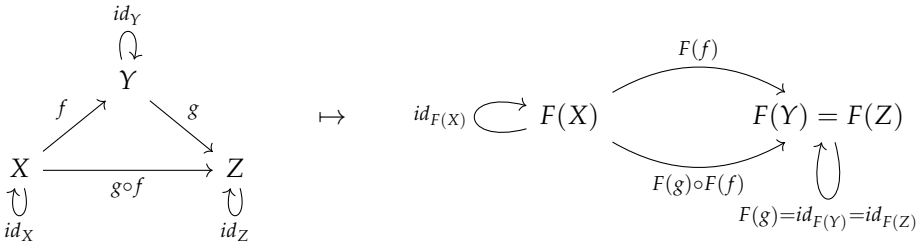
 doi:10.20944/preprints202403.1479.v1

while, in the second instance, imposing the additional algebraic equivalences $F(W) = F(Z)$ and $F(Y) = F(Z)$ on objects in the codomain category yields instead a fibration over a two-object, eight-morphism base category:



(104)

These fibrations are still discrete, but now in the former case there are only two discrete fiber categories: one for object $F(X)$ in the base category $\mathcal{B}$, consisting of the single object X from the total category $\mathcal{E}$ (along with its identity morphism $id_X : X \to X$), and one for object $F(Y) = F(Z)$ in the base category $\mathcal{B}$, consisting of the pair of objects Y and Z from the total category $\mathcal{E}$ (along with their respective identity morphisms $id_Y : Y \to Y$ and $id_Z : Z \to Z$). In the latter case there are now also only two discrete fiber categories: one for object $F(X)$ in the base category $\mathcal{B}$, consisting of the single object X from the total category $\mathcal{E}$ (along with its identity morphism $id_X : X \to X$), and one for object $F(Y) = F(W) = F(Z)$ in the base category $\mathcal{B}$, consisting of the triple of objects Y, Z and W from the total category $\mathcal{E}$ (along with their respective identity morphisms $id_Y : Y \to Y$, $id_Z : Z \to Z$ and $id_W : W \to W$). Once again, these discrete fiber categories can be computed automatically, directly from the `AbstractFunctor` objects in CATEGORICA, as shown in Figure 26. Finally, we may consider applying, in the first instance, the additional algebraic equivalence $(F(g) : F(Y) \to F(Z)) = \left(id_{F(Z)} : F(Z) \to F(Z)\right)$ on morphisms in the codomain category, thus yielding the following fibration over a two-object, four-morphism base category:



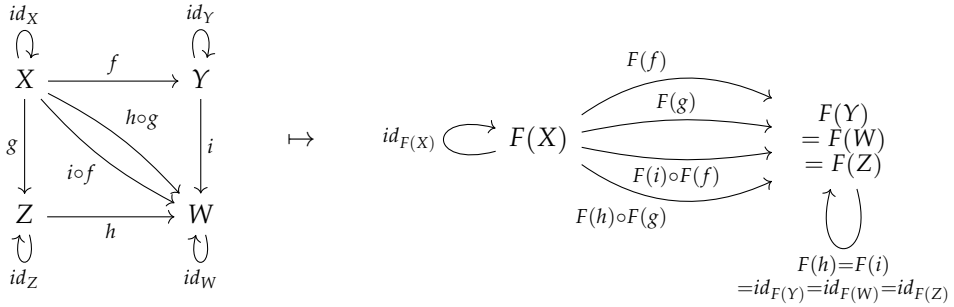
(105)

or alternatively, in the second instance, the additional algebraic equivalences:

$$(F(h) : F(Z) \to F(W)) = \left(id_{F(Z)} : F(Z) \to F(Z)\right), \quad \text{and} \quad (F(i) : F(Y) \to F(W)) = \left(id_{F(Z)} : F(Z) \to F(Z)\right), \tag{106}$$

on morphisms in the codomain category, thus yielding the following fibration over a two-object, six-morphism base category:

 doi:10.20944/preprints202403.1479.v1



(107)

The resulting fibrations are no longer discrete, since in the former case the fiber category for object $F(Y) = F(Z)$ in the base category $\mathcal{B}$ now contains a morphism $g : Y \to Z$ connecting the pair of objects Y and Z from the total category $\mathcal{E}$, and in the latter case the fiber category for object $F(Y) = F(W) = F(Z)$ in the base category $\mathcal{B}$ now contains a pair of morphisms $h : Z \to W$ and $i : Y \to W$ connecting the triple of objects Y, Z and W from the total category $\mathcal{E}$, as seen in the explicit CATEGORICA implementation in Figure 27. These examples make manifest the precise relationship between category-theoretic fibrations and topological ones: if one considers categories whose objects are points and whose morphisms are paths between those points, then the abstract definition of a Grothendieck fibration reduces concretely to the classical definition a fiber bundle, with the total category $\mathcal{E}$ being the total space, the base category $\mathcal{B}$ being the base space, and the fiber categories F_X being the fibers for each point X in the base.

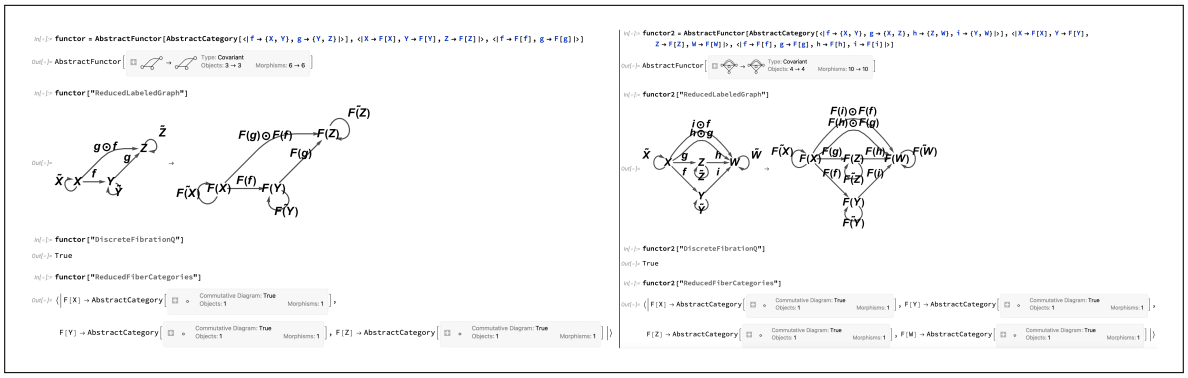


Figure 25. On the left, an `AbstractFunctor` object corresponding to a discrete fibration of a simple three-object, six-morphism total category over a three-object, six-morphism base category, showing the three discrete (single-object) fiber categories. On the right, an `AbstractFunctor` object corresponding to a discrete fibration of a slightly larger four-object, ten-morphism total category over a four-object, ten-morphism base category, showing the four discrete (single-object) fiber categories.

 doi:10.20944/preprints202403.1479.v1

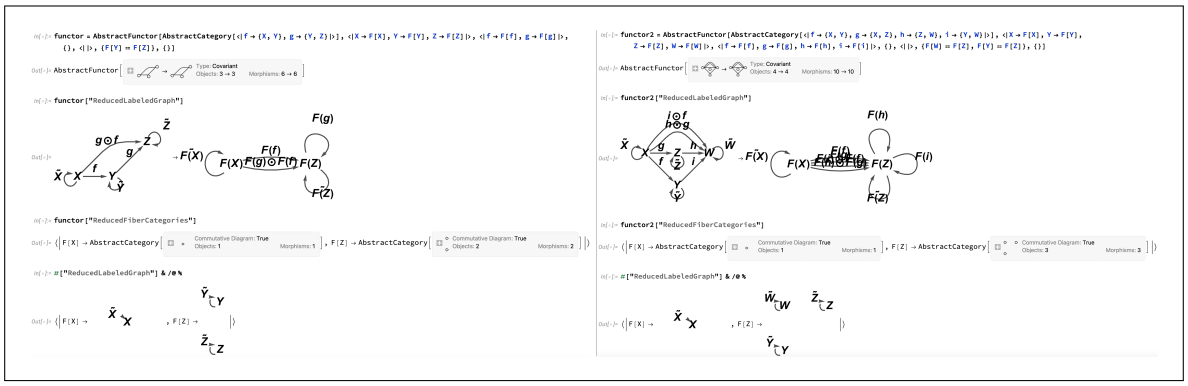


Figure 26. On the left, an `AbstractFunctor` object corresponding to a more general fibration of a simple three-object, six-morphism total category over a two-object, five-morphism base category, obtained by imposing the additional object equivalence $F(Y) = F(Z)$ in the codomain category, showing the two resulting discrete fiber categories. On the right, an `AbstractFunctor` object corresponding to a more general fibration of a slightly larger four-object, ten-morphism total category over a two-object, eight-morphism base category, obtained by impoosing the additional object equivalences $F(W) = F(Z)$ and $F(Y) = F(Z)$ in the codomain category, showing the two resulting discrete fiber categories.

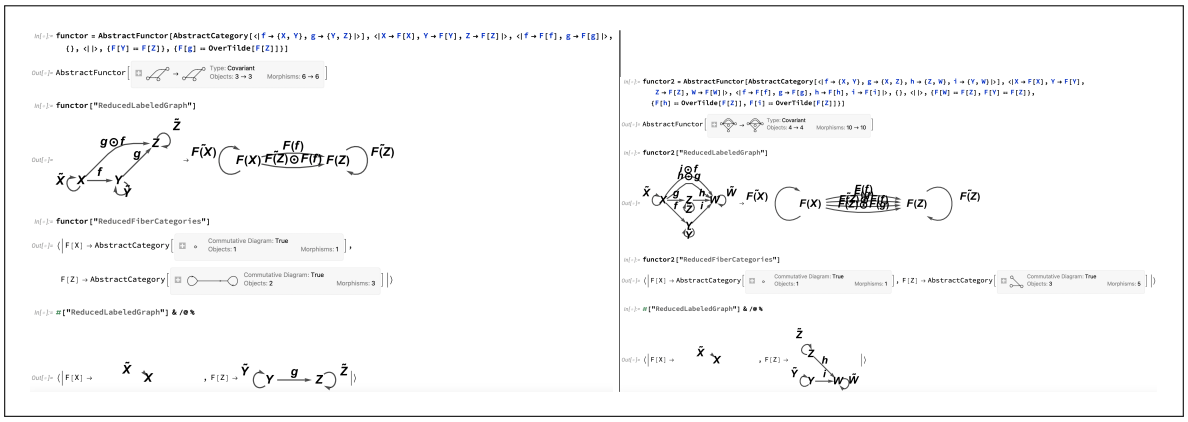


Figure 27. On the left, an `AbstractFunctor` object corresponding to a more general fibration of a simple three-object, six-morphism total category over a two-object, four-morphism base category, obtained by imposing the additional object equivalence $F(Y) = F(Z)$ and additional morphism equivalence $F(g) = id_{F(Z)}$ in the codomain category, showing the two resulting non-discrete fiber categories. On the right, an `AbstractFunctor` object corresponding to a more general fibration of a slightly larger four-object, ten-morphism total category over a two-object, six-morphism base category, obtained by imposing the additional object equivalences $F(W) = F(Z)$ and $F(Y) = F(Z)$, and additional morphism equivalences $F(h) = id_{F(Z)}$ and $F(i) = id_{F(Z)}$, in the codomain category, showing the two resulting non-discrete fiber categories.

There exist many other standard features and properties of functors that are currently supported within CATEGORICA, and that can be computed automatically using the `AbstractFunctor` function, including several that can be effectively assembled as "composites" of the various features/properties already discussed above. For instance, a functor yielding an *equivalence* of categories (detectable through the *"EquivalenceFunctorQ"* property) is one that is simultaneously fully faithful *and* essentially surjective; an *embedding* functor (detectable through the *"EmbeddingFunctorQ"* property) is one that is simultaneously faithful *and* injective on objects; and a *full* embedding functor (detectable through the *"FullEmbeddingFunctorQ"* property) is one that is both *fully faithful* and injective on objects. An *embedding* of a category $\mathcal{C}$ into a category $\mathcal{D}$ via the embedding functor $F : \mathcal{C} \to \mathcal{D}$ may be thought

 doi:10.20944/preprints202403.1479.v1

of as being a weaker/more lax version of an *inclusion* of a *subcategory* $\mathcal{C}$ into a *supercategory* $\mathcal{D}$, i.e. a functor $F : \mathcal{C} \to \mathcal{D}$ such that the object and morphism sets of the domain category $\mathcal{C}$ form (potentially improper) subsets of the respective object and morphism sets of the codomain category $\mathcal{D}$:

$$\text{ob}(\mathcal{C}) \subseteq \text{ob}(\mathcal{D}), \qquad \text{and} \qquad \text{hom}(\mathcal{C}) \subseteq \text{hom}(\mathcal{D}), \tag{108}$$

just as an *equivalence* between categories $\mathcal{C}$ and $\mathcal{D}$ may be thought of as being a weaker/more lax version of a strict *equality* of categories (wherein the object and morphism sets of categories $\mathcal{C}$ and $\mathcal{D}$ are required to be strictly identical). Indeed, strict inclusions of subcategories into supercategories are also detectable, by means of the *"InclusionFunctorQ"* property in `AbstractFunctor` (along with the slightly stronger case in which the inclusion functor is also required to be full, detectable through the *"FullInclusionFunctorQ"* property). Other common types of functors that CATEGORICA contains specialized functionality for handling as part of the `AbstractFunctor` framework include *endofunctors* (i.e. functors mapping a domain category $\mathcal{C}$ to itself) via the *"EndofunctorQ"* property, *identity functors* (i.e. a trivial, much stronger case of an endofunctor in which every object and morphism in the domain category $\mathcal{C}$ is mapped to itself) via the *"IdentityFunctorQ"* property, *constant functors* (i.e. functors mapping every object in the domain category $\mathcal{C}$ to some fixed object X in the codomain category $\mathcal{D}$, and mapping every morphism in $\mathcal{C}$ to the identity morphism $id_X : X \to X$ on that object) via the *"ConstantFunctorQ"* property, and *conservative functors* (i.e. functors for which a given morphism $F(f) : F(X) \to F(Y)$ in the codomain category $\mathcal{D}$ being an isomorphism necessarily implies that the corresponding morphism $f : X \to Y$ in the domain category $\mathcal{C}$ must also have been an isomorphism) via the *"ConservativeFunctorQ"* property, etc.

5. Natural Transformations and Algorithmic Details

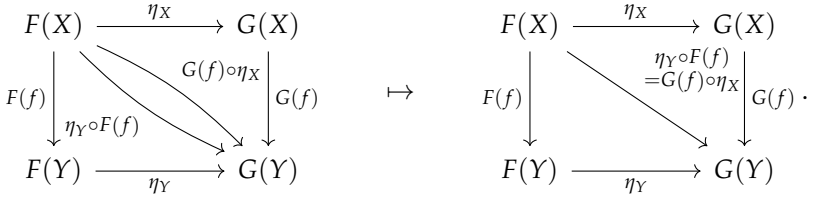
Having considered above the case of homomorphisms/structure-preserving maps between categories, in the form of functors, it is now possible to consider the case of homomorphisms/structure-preserving maps between *functors*, in the form of *natural transformations*. More precisely, if $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ are both functors mapping between the same pair of domain/codomain categories $\mathcal{C}$ and $\mathcal{D}$, then the map $\eta : F \Rightarrow G$ is a natural transformation if and only if it associates every object X in category $\mathcal{C}$ to a corresponding morphism $\eta_X : F(X) \to G(X)$ in category $\mathcal{D}$:

$$\forall X \in \text{ob}(\mathcal{C}), \qquad (\eta_X : F(X) \to G(X)) \in \text{hom}(\mathcal{D}), \tag{109}$$

known as the *component* of the natural transformation η at X, in such a way that, for every morphism $f : X \to Y$ in category $\mathcal{C}$, one has that the composition $\eta_Y \circ F(f)$ yields the same morphism as the composition $G(f) \circ \eta_X$ in category $\mathcal{D}$, i.e:

$$\forall (f : X \to Y) \in \text{hom}(\mathcal{C}), \qquad (\eta_Y \circ F(f) : F(X) \to G(Y)) = (G(f) \circ \eta_X : F(X) \to G(Y)), \tag{110}$$

which can be illustrated simply by means of the following commutative diagram (which therefore commutes in category $\mathcal{D}$):



(111)

Natural transformations abstract and generalize many powerful results and constructions in mathematics, especially in algebraic topology, perhaps most notably the Hurewicz theorem/Hurewicz homomorphism relating homotopy groups and homology groups of spaces: both the construction of

 doi:10.20944/preprints202403.1479.v1

homotopy groups and the construction of homology groups over a (pointed) topological space may be formalized as functors from the category $\mathbf{Top}_*$ of pointed topological spaces and continuous functions between them, to the category **Grp** of groups and group homomorphisms, with the Hurewicz homomorphism being elegantly represented as a natural transformation between these two functors (and the Hurewicz theorem corresponding to the statement that such a natural transformation necessarily exists). In linear algebra and functional analysis, the relationship between the double-dual V^{**} of a vector space V and the original space constitutes another familiar example of a commonplace natural transformation in mathematics; the double-dual operation may be formalized as an endofunctor on the category **Vect** of vector spaces and linear maps (i.e. a functor from **Vect** to itself), and there necessarily exists a natural transformation between this double-dual functor and the identity functor on **Vect**. For the case of finite-dimensional vector spaces, this natural transformation becomes a *natural isomorphism* (a stronger case, to be defined momentarily).

CATEGORICA is able to represent arbitrary natural transformations between `AbstractFunctor` objects using the `AbstractNaturalTransformation` function. In particular, it is able to compute the minimum set of algebraic equivalences between component morphisms that is necessary in order for a particular transformation between `AbstractFunctor` objects to be natural, and to represent these equivalences in both equational and (commutative) diagrammatic form, as illustrated in Figure 28 for the minimal case presented above, in which one considers a natural transformation between a pair of functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ acting on a common domain category $\mathcal{C}$ consisting of a pair of objects X and Y and a single morphism $f : X \to Y$, i.e. one has:

$$id_X \circlearrowright X \xrightarrow{f} Y \circlearrowright id_Y \qquad \mapsto \qquad id_{F(X)} \circlearrowright F(X) \xrightarrow{F(f)} F(Y) \circlearrowright id_{F(Y)} \,, \tag{112}$$

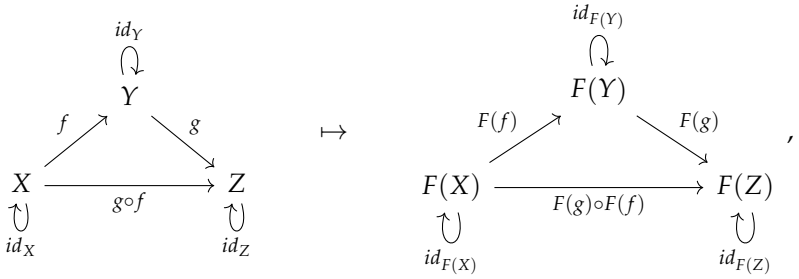
and:

$$id_X \circlearrowright X \xrightarrow{f} Y \circlearrowright id_Y \qquad \mapsto \qquad id_{G(X)} \circlearrowright G(X) \xrightarrow{G(f)} G(Y) \circlearrowright id_{G(Y)} \,, \tag{113}$$

respectively, in which the minimum algebraic condition for naturality is precisely the one given above:

$$(\eta_Y \circ F(f) : F(X) \to G(Y)) = (G(f) \circ \eta_X : F(X) \to G(Y)), \tag{114}$$

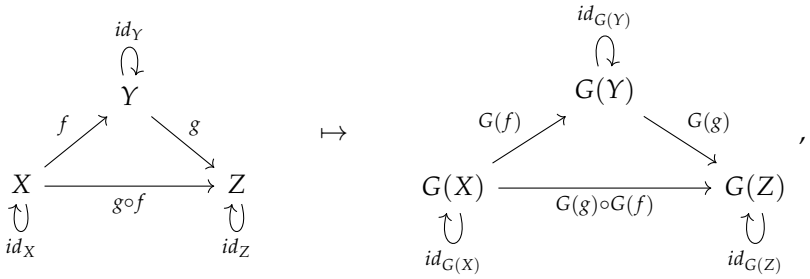
plus a couple of corollary conditions on the identity morphisms. However, although this particular example is especially simple, one does not need to increase the size or complexity of the underlying categories and functors involved very much before the resulting naturality conditions become rather unwieldy to manipulate by hand. For instance, let us consider the next most obvious example, in which the common domain category $\mathcal{C}$ of the functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ consists instead of three objects X, Y and Z, along with a pair of composable morphisms $f : X \to Y$ and $g : Y \to Z$, i.e. one now has:



(115)

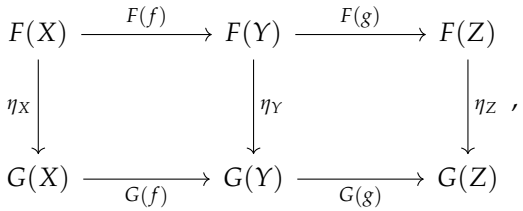
and:

 doi:10.20944/preprints202403.1479.v1



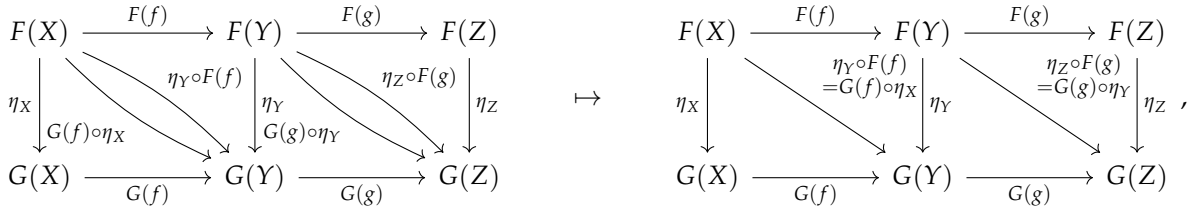
(116)

respectively. In this case, the relevant commutative diagram characterizing the natural transformation $\eta : F \Rightarrow G$ becomes instead the commutative oblong:



(117)

wherein the two interior squares must both commute:

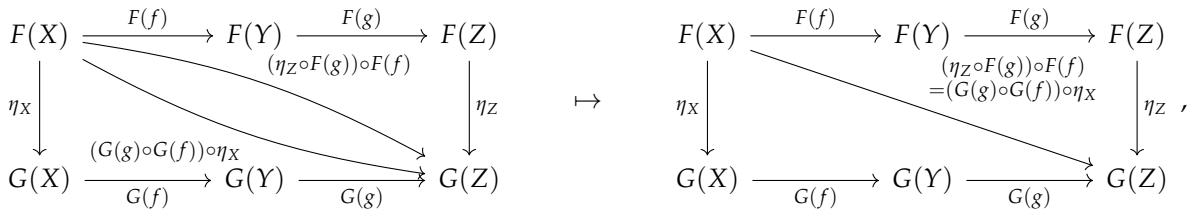


(118)

i.e:

$$(\eta_Y \circ F(f) : F(X) \to G(Y)) = (G(f) \circ \eta_X : F(X) \to G(Y)),$$
$$\text{and} \qquad (\eta_Z \circ F(g) : F(Y) \to G(Z)) = (G(g) \circ \eta_Y : F(Y) \to G(Z)), \tag{119}$$

in addition to the outer rectangle:



(120)

i.e:

$$((\eta_Z \circ F(g)) \circ F(f) : F(X) \to G(Z)) = ((G(g) \circ G(f)) \circ \eta_X : F(X) \to G(Z)), \tag{121}$$

plus all corresponding conditions on the identity morphisms. As shown in Figure 29, CATEGORICA is nevertheless easily able to represent, and to perform automatic computations involving, these more complex cases of natural transformations too.

 doi:10.20944/preprints202403.1479.v1

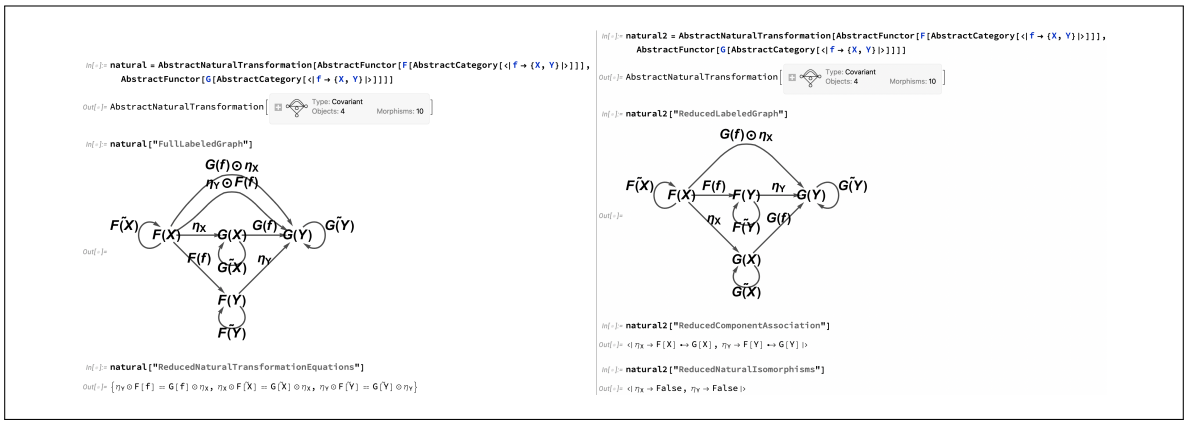


Figure 28. On the left, the `AbstractNaturalTransformation` object for a pair of functors $F, G : \mathcal{C} \to \mathcal{D}$ between two very simple (two-object, three-morphism) categories, showing the minimum set of algebraic equivalences on the component morphisms necessary for the natural transformation to be valid. On the right, the same `AbstractNaturalTransformation` object for the pair of functors $F, G : \mathcal{C} \to \mathcal{D}$ between the same two very simple (two-object, three-morphism) categories, showing the labeled graph representation with the aforementioned algebraic equivalences imposed, as well as the association of component morphisms, showing that neither component is a natural isomorphism.

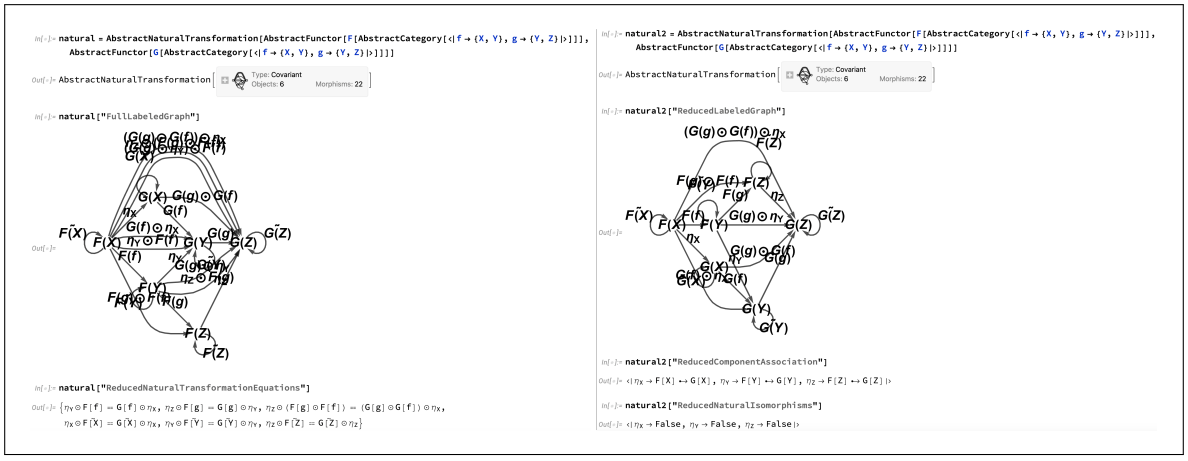
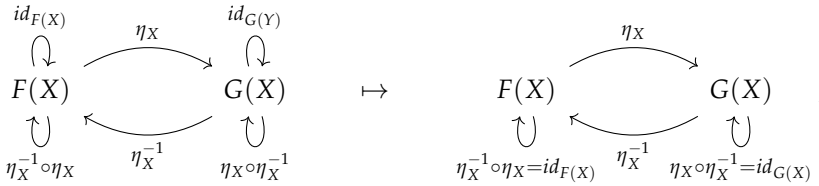


Figure 29. On the left, the `AbstractNaturalTransformation` object for a pair of functors $F, G : \mathcal{C} \to \mathcal{D}$ between two slightly more complex (three-object, six-morphism) categories, showing the minimum set of algebraic equivalences on the component morphisms necessary for the natural transformation to be valid. On the right, the same `AbstractNaturalTransformation` objects for the pair of functors $F, G : \mathcal{C} \to \mathcal{D}$ between the same two slightly more complex (three-object, six-morphism) categories, showing the labeled graph representation with the aforementioned algebraic equivalences imposed, as well as the association of component morphisms, showing that none of the components is a natural isomorphism.

Figures 28 and 29 also demonstrate CATEGORICA's ability to detect *natural isomorphisms*: components $\eta_X : F(X) \to G(X)$ of the natural transformation $\eta : F \Rightarrow G$ that are also isomorphisms, i.e. where the inverse morphism $\eta_X^{-1} : G(X) \to F(X)$ also exists in the codomain category $\mathcal{D}$, such that $\eta_X^{-1} \circ \eta_X : F(X) \to F(X)$ is the identity morphism on $F(X)$ and $\eta_X \circ \eta_X^{-1} : G(X) \to G(X)$ is the identity morphism on $G(X)$:

 doi:10.20944/preprints202403.1479.v1



(122)

i.e:

$$\begin{aligned}
&\forall X \in \mathrm{ob}(\mathcal{C}), \qquad \text{component } \eta_X : F(X) \to G(X) \qquad \text{of} \qquad \eta : (F : \mathcal{C} \to \mathcal{D}) \Rightarrow (G : \mathcal{C} \to \mathcal{D}),\\
&\qquad \text{is a natural isomorphism} \qquad \Longleftrightarrow \qquad \exists \Big(\eta_X^{-1} : G(X) \to F(X)\Big) \in \mathrm{hom}(\mathcal{D}),\\
&\qquad\qquad \text{such that} \qquad \Big(\eta_X^{-1} \circ \eta_X : F(X) \to F(X)\Big) = \Big(id_{F(X)} : F(X) \to F(X)\Big),\\
&\qquad\qquad\qquad \text{and} \qquad \Big(\eta_X \circ \eta_X^{-1} : G(X) \to G(X)\Big) = \Big(id_{G(X)} : G(X) \to G(X)\Big).
\end{aligned} \tag{123}$$

In addition to individual objects in the codomain category $\mathcal{D}$ being naturally isomorphic, one can also speak of a pair of functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ themselves as being naturally isomorphic: if *all* components $\eta_X : F(X) \to G(X)$ are natural isomorphisms (for every object X in the domain category $\mathcal{C}$), then the natural transformation $\eta : F \Rightarrow G$ itself is known as a natural isomorphism between functors:

$$\begin{aligned}
&\eta : (F : \mathcal{C} \to \mathcal{D}) \Rightarrow (G : \mathcal{C} \to \mathcal{D}) \text{ is a natural isomorphism} \qquad \Longleftrightarrow \qquad \forall X \in \mathrm{ob}(\mathcal{C}),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{component } \eta_X : F(X) \to G(X) \text{ is a natural isomorphism.}
\end{aligned} \tag{124}$$

The intuition underlying the concept of natural isomorphisms between functors can be precisely formalized by considering the *functor category* $\mathcal{D}^{\mathcal{C}}$ (assuming that $\mathcal{C}$ and $\mathcal{D}$ are arbitrary small categories), whose objects are given by the functors $F : \mathcal{C} \to \mathcal{D}$ and whose morphisms are given by the natural transformations $\eta : F \Rightarrow G$ between such functors, wherein all isomorphisms between objects represent natural isomorphisms between functors. To determine whether a given `AbstractNaturalTransformation` object in CATEGORICA represents a natural isomorphism between functors, one can use the *"NaturalIsomorphismQ"* property. CATEGORICA is also able to determine whether the codomain categories $\mathcal{D}$ between the two functors actually match (via the *"MatchingCodomainsQ"* property), and therefore whether the `AbstractNaturalTransformation` object does indeed represent a valid natural transformation (via the *"ValidNaturalTransformationQ"* property), meaning *both* that the two codomain categories match *and* that the defining algebraic conditions of a natural transformation hold within those codomain categories.

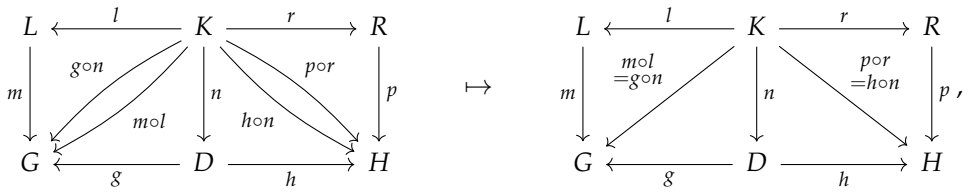
Finally, it is worth noting that all of CATEGORICA's core algebraic and diagrammatic reasoning algorithms are represented internally in terms of (hyper)graph rewriting systems over labeled graph representations of quivers and categories, whereby one formalizes (hyper)graph rewriting rules as *spans* of monomorphisms [31,32]:

$$L \xleftarrow{\ l\ } K \xrightarrow{\ r\ } R\ , \tag{125}$$

i.e. pairs of monomorphisms which share a common domain object K. Here, the ambient category $\mathcal{C}$ in which these monomorphisms exist is the category whose objects are (hyper)graphs and whose morphisms are (hyper)graph inclusion relations. In the above, object L represents the (hyper)graph pattern appearing on the left-hand/input side of the rule, object R represents the (hyper)graph pattern appearing on the right-hand/output side of the rule (i.e. the pattern intended to replace pattern L wherever it appears), and object K represents the "residual" sub(hyper)graph pattern shared by both

 doi:10.20944/preprints202403.1479.v1

patterns L and R that remains invariant after the left-hand-side L pattern has been "extracted", but before the right-hand-side pattern R has been "injected". Such a rewriting rule may then be said to *match* a given (hyper)graph object G if there exists a morphism $m : L \to G$ in the category $\mathcal{C}$. The resulting (hyper)graph H obtained via application of the rewriting rule at this particular match may consequently be computed explicitly by completing the following *double-pushout* diagram [33]:

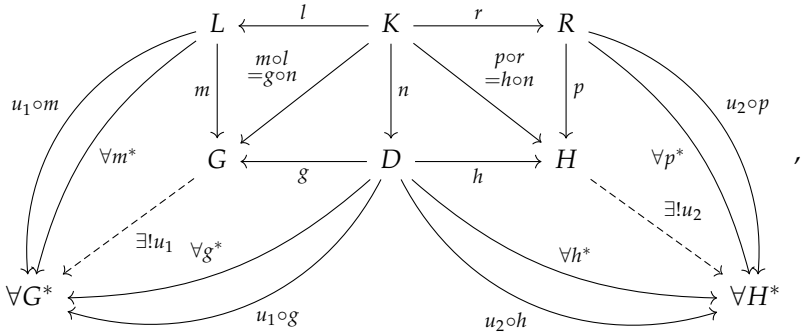


(126)

i.e. one must proceed to find objects D and H, and morphisms n, p, g and h, in the category $\mathcal{C}$, such that the diagram above commutes:

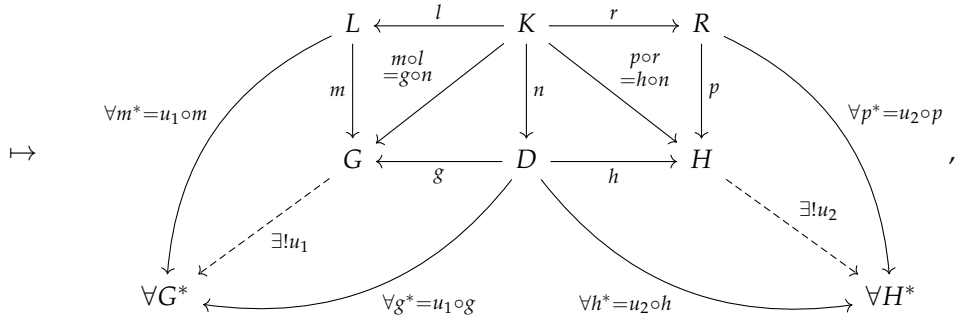
$$\begin{aligned} &\exists D, H \in \mathrm{ob}(\mathcal{C}), \qquad \exists (n : L \to D), (p : R \to H), (g : D \to G), (h : D \to H) \in \mathrm{hom}(\mathcal{C}), \\ &\qquad \text{such that} \qquad (m \circ l : K \to G) = (g \circ n : K \to G), \\ &\qquad\qquad \text{and} \qquad (p \circ r : K \to H) = (h \circ n : K \to H). \end{aligned} \tag{127}$$

Furthermore, these objects and morphisms (and therefore, in particular, the object H representing the resulting (hyper)graph) are guaranteed to be unique, up to natural isomorphism, by virtue of the following *universal property* of the double-pushout diagram:



(128)

which then collapses down to:



(129)

i.e. for any other objects G^* and H^*, and morphisms m^*, g^*, p^* and h^*, in the category $\mathcal{C}$, such that the resulting squares commute:

 doi:10.20944/preprints202403.1479.v1

$$\forall G^*, H^* \in \mathrm{ob}(\mathcal{C}), \quad \forall (m^* : L \to G^*), (g^* : D \to G^*), (p^* : R \to H^*), (h^* : D \to H^*) \in \mathrm{hom}(\mathcal{C})$$
$$\text{such that} \quad (m^* \circ l : K \to G^*) = (g^* \circ n : K \to G^*),$$
$$\text{and} \quad (p^* \circ r : G \to H^*) = (h^* \circ n : K \to H^*), \quad (130)$$

there necessarily exists a pair of unique morphisms $u_1 : G \to G^*$ and $u_2 : H \to H^*$ such that the diagram above commutes, i.e:

$$\exists! (u_1 : G \to G^*), \quad \text{such that} \quad (m^* : L \to G^*) = (u_1 \circ m : L \to G^*),$$
$$\text{and} \quad (g^* : D \to G^*) = (u_1 \circ g : D \to G^*), \quad (131)$$

and:

$$\exists! (u_2 : H \to H^*), \quad \text{such that} \quad (p^* : R \to H^*) = (u_2 \circ p : R \to H^*),$$
$$\text{and} \quad (h^* : D \to H^*) = (u_2 \circ h : D \to H^*). \quad (132)$$

CATEGORICA is able to exploit the formalism of double-pushout (hyper)graph rewriting in order to convert automatically between potentially slow, tedious algebraic computations and fast, explicit diagrammatic ones. To this end, the core of CATEGORICA's hypergraph rewriting system has been built on top of the GRAVITAS computational general relativity framework [51,52] (which has previously been applied to the study of black hole physics [53–55] and algebraic quantum field theory [56]), which employs a bespoke suite of highly optimized, and massively parallelized, algorithms for rewriting large hypergraphs subject to generic algebraic constraints. We note also that such rewriting systems, especially in the *multiway*/non-deterministic case, admit an elegant mathematical description in terms of higher categories and (higher) homotopy types [57,58].

6. Concluding Remarks

In this article, we have intended to provide a reasonably comprehensive overview of several of the core data structures of the CATEGORICA framework, especially the foundational `AbstractQuiver`, `AbstractCategory`, `AbstractFunctor` and `AbstractNaturalTransformation` constructions, as well as a number of its core design principles; in particular, it is hoped that this overview will lay the groundwork for the forthcoming second article in this series, which will showcase and outline CATEGORICA's more advanced functionality for handling algebraic computations and automated theorem-proving for abstract universal properties (e.g. products, coproducts, pullbacks, pushouts, limits and colimits), support for monoidal categories and computations involving string diagrams, preliminary support for elementary toposes, and initial support for higher categories. There exist many planned directions for future research and development involving the CATEGORICA framework, including the extension of the `AbstractFunctor` function to support the detection and handling of Grothendieck fibrations in complete generality, the inclusion of adjoint functor relationships between `AbstractFunctor` objects (and therefore support for weak equivalences between `AbstractCategory` objects), and the incorporation of full support for both strict and weak 2-categories (i.e. bicategories) by extending the current support for single-object bicategories via the `AbstractStrictMonoidalCategory` functionality [59]. Similarly, all instances of `AbstractCategory` objects in the CATEGORICA framework are currently assumed by default to be enriched over **Set** (since the object sets and morphism sets are assumed to be pure Wolfram Language list structures); it would therefore be instructive to extend the present functionality to accommodate `AbstractCategory` objects enriched directly over arbitrary `AbstractStrictMonoidalCategory` objects [60,61] . Indeed, although there exists much preliminary

 doi:10.20944/preprints202403.1479.v1

infrastructural support for general monoidal categories, there are also many important cases and constructions (including braided monoidal categories [62,63], monoidal functors [64], etc.) which are used throughout the field of applied category, and which are not currently supported, although appropriate functionality for many of these is presently under active development.

Acknowledgments: The author would like to thank participants in the GReTA - Graph Transformation Theory and Applications - seminar series (especially Nicolas Behr and Aleks Kissinger), and members of the Topos Institute (especially Owen Lynch and David Spivak) for various discussions and comments on aspects of the design and implementation of the CATEGORICA framework. The author would also like to thank Mohamed Barakat, Manojna Namuduri and Stephen Wolfram for their encouragement and design input at several stages of the development process.

References

1. S. EILENBERG AND S. MAC LANE (1942), "Group Extensions and Homology", *Annals of Mathematics* **43** (4): 757–831. https://www.jstor.org/stable/1968966.
2. S. EILENBERG AND S. MAC LANE (1945), "General Theory of Natural Equivalences", *Transactions of the American Mathematical Society* **58** (2): 231–294. https://www.jstor.org/stable/1990284.
3. J-P. SERRE (1955), "Faisceaux Algebriques Coherents", *Annals of Mathematics* **61** (2): 197–278. https://www.jstor.org/stable/1969915.
4. A. GROTHENDIECK (1957), "Sur quelques points d'algèbre homologique, I", *Tôhoku Mathematical Journal* **9** (2): 119–221. https://projecteuclid.org/journals/tohoku-mathematical-journal/volume-9/issue-2/Sur-quelques-points-dalgÃÍbre-homologique-I/10.2748/tmj/1178244839.full.
5. C. MCLARTY (1996), *Elementary Categories, Elementary Toposes*, Oxford Logic Guides **21**. Clarendon Press, Oxford Science Publications. ISBN: 978-0198514732.
6. P. T. JOHNSTONE (2002), *Sketches of an Elephant: A Topos Theory Compendium* (Volumes 1 and 2), Oxford Logic Guides **43** and **44**. Clarendon Press, Oxford Science Publications. ISBN: 978-0198534259/978-0198515982.
7. K. KUNEN (1983), *Set Theory: An Introduction to Independence Proofs* (Reprint Edition), Studies in Logic and the Foundations of Mathematics **102**. North Holland. ISBN: 978-0444868398.
8. E. RIEHL (2016), *Category Theory in Context*, Aurora: Dover Modern Math Originals. Dover Publications. ISBN: 978-0486809038.
9. P. A. M. DIRAC (1982), *The Principles of Quantum Mechanics* (Fourth Edition), International Series of Monographs on Physics **27**. Clarendon Press, Oxford Science Publications. ISBN: 978-0198520115.
10. J. VON NEUMANN (1955), *Mathematical Foundations of Quantum Mechanics* (First Edition), Princeton Landmarks in Mathematics. Princeton University Press. ISBN: 978-0691028934.
11. S. ABRAMSKY AND B. COECKE (2004), "A categorical semantics of quantum protocols", *Proceedings of the 19th IEEE Symposium on Logic in Computer Science*. Turku, Finland: 415–425. https://arxiv.org/abs/quant-ph/0402130.
12. S. ABRAMSKY AND B. COECKE (2008), "Categorical quantum mechanics", *Handbook of Quantum Logic and Quantum Structures*, K. Engesser, D. M. Gabbay, D. Lehmann (eds): 261–323. Elsevier. https://arxiv.org/abs/0808.1023.
13. B. COECKE AND R. DUNCAN (2008), "Interacting Quantum Observables", *International Colloquium on Automata, Languages and Progamming*, Lecture Notes in Computer Science **5126**: 298–310. Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-540-70583-3_25.
14. B. COECKE AND R. DUNCAN (2009), "Interacting Quantum Observables: Categorical Algebra and Diagrammatics", *New Journal of Physics* **13** (4): 043016. https://arxiv.org/abs/0906.4725.
15. B. COECKE, M. SADRZADEH AND S. CLARK (2010), "Mathematical Foundations for a Compositional Distribution Model of Meaning", *Linguistic Analysis* **36** (Lambek Festschrift): 345–384. https://arxiv.org/abs/1003.4394.
16. M. CAPUCCI, B. GAVRANOVIĆ, J. HEDGES AND E. F. RISCHEL (2021), "Towards Foundations of Categorical Cybernetics", *Proceedings of Applied Category Theory 2021, Electronic Proceedings in Theoretical Computer Science* **372**: 235–248. https://arxiv.org/abs/2105.06332v2.
17. G. S. H. CRUTTWELL, B. GAVRANOVIĆ, N. GHANI, P. WILSON AND F. ZANASI (2022), "Categorical Foundations of Gradient-Based Learning", *European Symposium on Programming 2022*, Lecture Notes in Computer Science **13240**: 1–28. Springer, Cham. https://arxiv.org/abs/2103.01931.

 doi:10.20944/preprints202403.1479.v1

18. J. C. Baez and B. S. Pollard (2017), "A Compositional Framework for Reaction Networks", *Reviews in Mathematical Physics* **29** (9): 1750028. https://arxiv.org/abs/1704.02051.
19. J. C. Baez and J. Master (2020), "Open Petri Nets", *Mathematical Structures in Computer Science* **30** (3): 314–341. https://arxiv.org/abs/1808.05415v4.
20. J. Gorard (2022), "A Functorial Perspective on (Multi)computational Irreducibility", *arXiv preprint*: https://arxiv.org/abs/2301.04690.
21. J. Gorard (2018), "The Slowdown Theorem: A Lower Bound for Computational Irreducibility in Physical Systems", *Complex Systems* **27** (2): 177–185. https://www.complex-systems.com/abstracts/v27_i02_a05/.
22. D. I. Spivak (2012), "Functorial Data Migration", *Information and Computation* **217**: 31–51. https://arxiv.org/abs/1009.1166.
23. B. Fong and D. I. Spivak (2019), *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press. ISBN: 978-1108711821. https://arxiv.org/abs/1803.05316.
24. M. Halter, E. Patterson, A. Baas and J. Fairbanks (2020), "Compositional Scientific Computing with Catlab and SemanticModels", *arXiv preprint*: https://arxiv.org/abs/2005.04831.
25. A. Kissinger and V. Zamdhiev (2015), "Quantomatic: A Proof Assistant for Diagrammatic Reasoning", *International Conference on Automated Deduction - CADE-25*: 326–336. https://arxiv.org/abs/1503.01034.
26. P. Sobociński, P. W. Wilson and F. Zanasi (2019), "Cartographer: A Tool for String Diagrammatic Reasoning (Tool Papear), *8th Conference on Algebra and Coalgebra in Computer Science - CALCO 2019*: 20:1–20:7. https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CALCO.2019.20.
27. A. Kissinger and J. van de Wetering (2019), "PyZX: Large Scale Automated Diagrammatic Reasoning", *16th International Conference on Quantum Physics and Logic* **318**: 229–241. https://arxiv.org/abs/1904.04735.
28. J. Gorard (2020), "Some Relativistic and Gravitational Properties of the Wolfram Model", *Complex Systems* **29** (2): 599–654. https://arxiv.org/abs/2004.14810.
29. J. Gorard (2020), "Some Quantum Mechanical Properties of the Wolfram Model", *Complex Systems* **29** (2): 537–598. https://www.complex-systems.com/abstracts/v29_i02_a02/.
30. J. Gorard (2020), "Algorithmic Causal Sets and the Wolfram Model", *arXiv preprint*: https://arxiv.org/abs/2011.12174.
31. E. Ehrig, M. Pfender and H. J. Schneider (1973), "Graph-grammars: An algebraic approach", *IEEE Conference Record of the 14th Annual Symposium on Switching and Automata Theory*: 167–180. https://ieeexplore.ieee.org/document/4569741.
32. H. Ehrig, K. Ehrig, U. Prange and G. Taentzer (2006), *Fundamentals of Algebraic Graph Transformation*, Monographs in Theoretical Computer Science. An EATCS Series. Berlin, Heidelberg: Springer. ISBN: 978-3642068317.
33. A. Habel, J. Müller and D. Plump (2001), "Double-pushout graph transformation revisited", *Mathematical Structures in Computer Science* **11** (5): 637–688. https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/doublepushout-graph-transformation-revisited/AF02050525390437E1DF746DE4459926.
34. J. Gorard, M. Namuduri and X. D. Arsiwalla (2020), "ZX-Calculus and Extended Hypergraph Rewriting Systems I: A Multiway Approach to Categorical Quantum Information Theory", *arXiv preprint*: https://arxiv.org/abs/2010.02752.
35. J. Gorard, M. Namuduri and X. D. Arsiwalla (2021), "ZX-Calculus and Extended Wolfram Model Systems II: Fast Diagrammatic Reasoning with an Application to Quantum Circuit Simplification", *arXiv preprint*: https://arxiv.org/abs/2103.15820.
36. J. Gorard, M. Namuduri and X. D. Arsiwalla (2021), "Fast Automated Reasoning over String Diagrams using Multiway Causal Structure", *arXiv preprint*: https://arxiv.org/abs/2105.04057.
37. J. Gorard (2016), "Uniqueness Trees: A Possible Polynomial Approach to the Graph Isomorphism Problem", *arXiv preprint*: https://arxiv.org/abs/1606.06399.
38. S. Mac Lane (1997), *Categories for the Working Mathematician* (Second Edition), Graduate Texts in Mathematics **5**. Berlin, Heidelberg: Springer. ISBN: 0-387-98403-8.
39. F. Borceux (2008), *Handbook of Categorical Algebra I: Basic Category Theory* (First Edition), Encyclopedia of Mathematics and its Applications **50**. Cambridge University Press. ISBN: 978-0521061193.
40. G. M. Bergman (2015), *An Invitation to General Algebra and Universal Constructions* (Second Edition). Springer, Cham. ISBN: 978-3319114774.

41. B. Mitchell (1965), *Theory of Categories* (First Edition), Pure and Applied Mathematics **17**. Academic Press. ISBN: 978-0124992504.
42. K. Borsuk (1947), "On The Topology of Retracts", *Annals of Mathematics* **48** (4): 1082–1094. https://www.jstor.org/stable/1969394.
43. S. Eilenberg and J. C. Moore (1965), *Foundations of Relative Homological Algebra*, Memoirs of the American Mathematical Society **55**. American Mathematical Society, Providence, Rhode Island. ISBN: 978-1470400019.
44. W. Dicks and E. Ventura (1996), *The Group Fixed by a Family of Injective Endomorphisms of a Free Group*, Contemporary Mathematics **195**. American Mathematical Society. ISBN: 978-0821877869.
45. R. Brown (2006), *Topology and Groupoids*. Booksurge Publishing. ISBN: 978-1419627224.
46. M. C. Pedicchio and W. Tholen (eds) (2003), *Categorical Foundations: Special Topics in Order, Topology, Algebra, and Sheaf Theory* (First Edition), Encyclopedia of Mathematics and its Applications **97**. Cambridge University Press. ISBN: 978-0521834148.
47. B. Pareigis (1971), *Categories and Functors* (First Edition), Pure and Applied Mathematics **39**. Academic Press. ISBN: 978-0125451505.
48. N. Jacobson (2009), *Basic Algebra II* (Second Edition), Dover Books on Mathematics. Dover Publications. ISBN: 978-0486471877.
49. A. Grothendieck (1959), "Technique de descente et théorèmes d'existence en géométrie algébrique. I. Généralités. Descente par morphismes fidèlement plats", *Séminaire Bourbaki* **5** (Exposé 190): 299–327. http://www.numdam.org/item/SB_1958-1960__5__299_0/.
50. J. W. Gray (1966), "Fibred and Cofibred Categories", *Proceedings of the Conference on Categorical Algebra*, S. Eilenberg, D. K. Harrison, S. Mac Lane, H. Röhrl (eds): 21–83. Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-642-99902-4_2.
51. J. Gorard (2023), "Computational General Relativity in the Wolfram Language using Gravitas I: Symbolic and Analytic Computation", *arXiv preprint*: https://arxiv.org/abs/2308.07508.
52. J. Gorard (2024), "Computational General Relativity in the Wolfram Language using Gravitas II: ADM Formalism and Numerical Relativity", *arXiv preprint*: https://arxiv.org/abs/2401.14209.
53. J. Gorard (2021), "Hypergraph Discretization of the Cauchy Problem in General Relativity via Wolfram Model Evolution", *arXiv preprint*: https://arxiv.org/abs/2102.09363.
54. J. Gorard (2023), "Non-Vacuum Solutions, Gravitational Collapse and Discrete Singularity Theorems in Wolfram Model Systems", *arXiv preprint*: https://arxiv.org/abs/2303.07282.
55. J. Gorard (2024), "General Relativistic Hydrodynamics in Discrete Spacetime: Perfect Fluid Accretion onto Static and Spinning Black Holes", *arXiv preprint*: https://arxiv.org/abs/2402.02331.
56. J. Gorard and J. Dannemann-Freitag (2023), "Axiomatic Quantum Field Theory in Discrete Spacetime via Multiway Causal Structure: The Case of Entanglement Entropies", *arXiv preprint*: https://arxiv.org/abs/2301.12455.
57. X. D. Arsiwalla, J. Gorard and H. Elshatlawy (2021), "Homotopies in Multiway (Non-Deterministic) Rewriting Systems as n-Fold Categories", *arXiv preprint*: https://arxiv.org/abs/2105.10822.
58. X. D. Arsiwalla and J. Gorard (2021), "Pregeometric Spaces from Wolfram Model Rewriting Systems as Homotopy Types", *arXiv preprint*: https://arxiv.org/abs/2111.03460.
59. S. Mac Lane (1963), "Natural Associativity and Commutativity", *Rice Institute Pamphlet - Rice University Studies* **49** (4): 28–46. https://repository.rice.edu/items/055560bd-a742-4571-b34e-c4829c92da16.
60. G. M. Kelly (1964), "On MacLane's conditions for coherence of natural associativities, commutativities, etc.", *Journal of Algebra* **1** (4): 397–402. https://www.sciencedirect.com/science/article/pii/0021869364900183
61. G. M. Kelly (1982), *Basic Concepts of Enriched Category Theory*, London Mathematical Society Lecture Notes Series. Cambridge: Cambridge University Press. ISBN: 978-0521287029.
62. A. Joyal and R. Street (1986), "Braided monoidal categories", *Macquarie Mathematics Reports* (860081). http://web.science.mq.edu.au/~street/JS1.pdf.
63. V. Chari and A. Pressley (1995), *A Guide to Quantum Groups*. Cambridge University Press. ISBN: 978-0521558846.
64. M. Aguiar and S. Mahajan (2010), *Monoidal Functors, Species and Hopf Algebras*, CRM Monograph Series **29**. American Mathematical Society, Ann Arbor, Michigan. ISBN: 978-0821847763.

 doi:10.20944/preprints202403.1479.v1