[13] J. -P. Serre, Structure de certains pro-p-groups, *Sem. Bourbaki 1962/63*, n. **252**.
[14] J. -P. Serre, Cohomologie Galoisienne, "Lectures Notes in Math." n. 5, Springer-Verlag, Berlin and New York, 1964.
[15] J. R. Stallings, Homology and central series of groups, *J. Algebra.* **2** (1965), 170–181.
[16] R. G. Swan, Groups of cohomological dimension one, *J. Algebra.* **12** (1969), 585–610.
[17] C. T. C. Wall, Poincaré complexes—I, *Ann. of Math.* **86** (1967), 213–245.

AMS 57B10

DEPARTMENT OF MATHEMATICS
THE GRADUATE SCHOOL
CITY UNIVERSITY OF NEW YORK
NEW YORK, NEW YORK

Properties Invariant within Equivalence Types of Categories

PETER FREYD

All of us know that any "mathematically relevant" property on categories is invariant within equivalence types of categories. Furthermore, we all know that any "mathematically relevant" property on objects and maps is preserved and reflected by equivalence functors. An obvious problem arises: How can we conveniently characterize such properties? The problem is complicated by the fact that the second mentioned piece of common knowledge, that equivalence functors preserve and reflect relevant properties on objects and maps, is just plain wrong.

I first met Sammy in the fall of 1958 and within ten minutes he was selling me on a "stylistic" point that turns out to be the central clue to the problem. (How often Sammy's "stylistic" points have totally changed entire mathematical viewpoints!) It took me 16 years to make the connection.

An equivalence $T: \mathbb{A} \to \mathbb{B}$ preserves equalizers but does not reflect them. $T(x)$ can be an equalizer of $T(y)$ and $T(z)$ without x being an equalizer of y and z, albeit for the most perverse of reasons, namely that the sources and targets of x, y, and z do not match as they should in $\mathbb{A}$ (since T can identify objects, they can match in $\mathbb{B}$).

To make the above stated problem amenable, I will restrict attention to elementary sentences in the language of categories, that is, sentences in which all quantifiers refer to objects and maps and the "atomic" predicates are compositions, equality, and source and target assertions. The standard approach to such a problem is to work with the "Frege notation"[1] (∀, ∃, ∧, ∨, etc.) and attempt an induction (not on sentences, but on formulas in general) on the number of bound variables. We cannot even begin here. Free formulas are not preserved by equivalence functors; in fact, none of the negations of atomic predicates are preserved by equivalence functors.

When I first met Sammy I was working on the metatheorem for abelian categories and he wanted me to state the metatheorem in a certain way. Note that none of us use the Frege notation very much. Note that we do write diagrams on the board and move our arms a bit. Sammy wanted me to formalize the latter. He was right. I must first describe a diagrammatic notation with which to solve the problem. (At the end, as it happens, we can translate back to the Frege notation. But only at the end.)

1. The Diagrammatic Language

By a **graph** I mean a collection of **vertices** together with a collection of **arrows**, each arrow assigned a **source vertex** and a **target vertex**. If one insists upon formalizing this in the standard set-theoretical way, then a graph is a quadruple $\langle V, A, s, t\rangle$, where s and t are functions from A to V.

Any category may be construed as a graph by forgetting compositions. Given a graph G and a category $\mathbb{A}$, a G-**diagram** in $\mathbb{A}$ is a graph homomorphism $D\colon G \to \mathbb{A}$. We could of course use the free category generated by a graph and turn everything into a discussion of functors. But a finite graph (e.g., one vertex, one arrow) can generate an infinite category (e.g., the monoid of natural numbers), and hence I stick to graphs.

A **path** in a graph is a finite word of arrows $\langle a_1, \ldots, a_n\rangle$ such that the target of a_i is the source of a_{i+1} for $i = 1, \ldots, n-1$. The source of the path is defined as the source of a_1 and the target of the path as the target of a_n. A **commutativity condition** on a graph is an ordered pair of paths each with the same source and target—unless one of the paths is empty, in which case we require that the source and target of the other be equal. A C-**Graph** is a graph together with a set of commutativity conditions. For a C-graph G, a G-diagram in $\mathbb{A}$ is a graph-homomorphism $D\colon G \to \mathbb{A}$ such that for every commutativity condition $\langle a_1, \ldots, a_n\rangle = \langle b_1, \ldots, b_m\rangle$ it is the case that $D(a_1)D(a_2)\cdots D(a_n) = D(b_1)\cdots D(b_m)$—unless $m = 0$, in which case we require that $D(a_1)\cdots D(a_n)$ be an identity map.

[1] Frege's notation, of course, was very different. The phrase "Frege notation," however, has come into standard use.

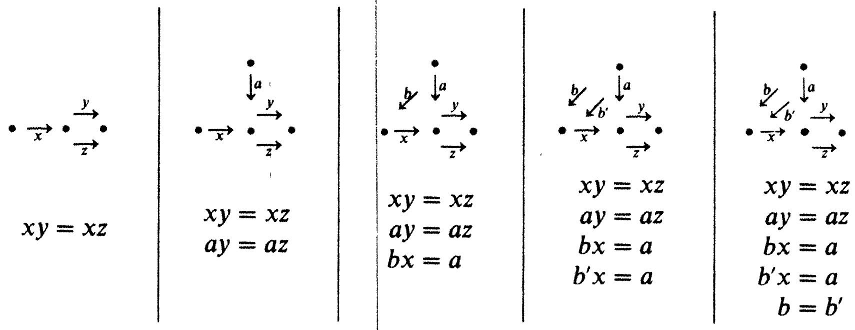

Fig. 1

Consider the nested family of C-graphs shown in Fig. 1. A diagram from the first is an equalizer diagram if and only if for every extension to the second there exists an extension to the third such that for every extension to the fourth there is an extension to the fifth.

A standard simplification of notation is to assume that every conceivable commutativity condition holds unless we say otherwise. I will say otherwise by inserting question marks within the graph, where it is to be understood that the question mark removes only one commutativity condition, namely that which immediately surrounds it. Figure 2 has three commutativity

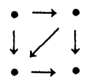
Fig. 2

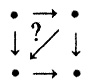
Fig. 3

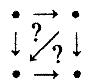
Fig. 4

conditions, Fig. 3 has two, and Fig. 4 still has one (the outer square). We may define a product diagram using Fig. 5. That is, a diagram from the first graph is a product if and only if for all extensions to the second

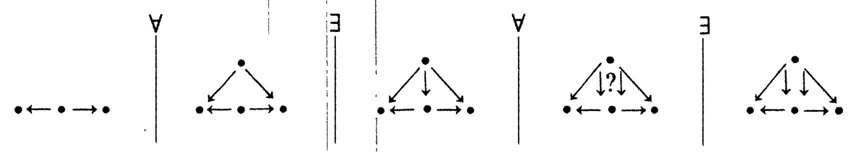
Fig. 5

there is an extension to the third such that for all extensions to the fourth there is an extension to the fifth. The last two C-diagrams are needed for the uniqueness, and when we arm-wave at a blackboard we customarily omit them and say the word "unique." Hence Fig. 6 defines product diagrams.

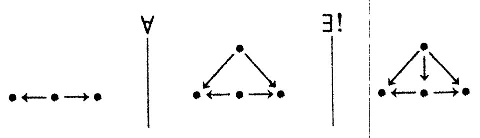

Fig. 6

A finite *rooted tree*, recall, is a finite partially ordered set with a unique smallest element (the *root*) such that two elements have a common upper bound if and only if one is less than the other. The immediate successors of the root will be called the *near-roots*; the tree that sprouts upwards from a near-root will be called its *corresponding subtree*.

A CG-**tree** is a finite rooted tree of C-graphs ordered by extension, each labeled by ∀ or ∃. We define the notion that a diagram $D: R \to \mathbb{A}$, where R is the root, **satisfies** the tree, recursively, as follows:

If the root is labeled ∃(∀), $D: R \to \mathbb{A}$ *satisfies the tree if an* (*if every*) *extension of D to a near-root satisfies the corresponding sub-CG-tree.*

If the tree is just its root, then $D: R \to \mathbb{A}$ satisfies the tree if and only if the root is labeled ∀.

If R is empty then the tree describes a property on categories, namely that the empty diagram satisfies the tree. For example, the linear tree of Fig. 7 is satisfied by $\varnothing \to \mathbb{A}$ if and only if $\mathbb{A}$ has binary products. We will call such properties **diagrammatic properties.**

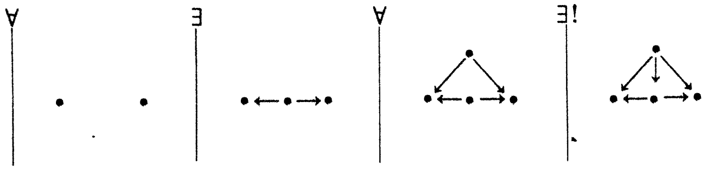

Fig. 7

Linear trees do not suffice. For example, the property that $\mathbb{A}$ is linearly connected requires a nonlinear tree such as that shown in Fig. 8. (One may check that a linear diagrammatic property is preserved under the formation of products of categories and that linear ordering is not so preserved.)

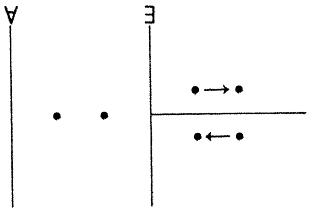

Fig. 8

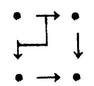

Fig. 9

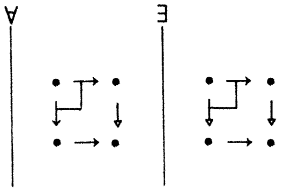

Fig. 10

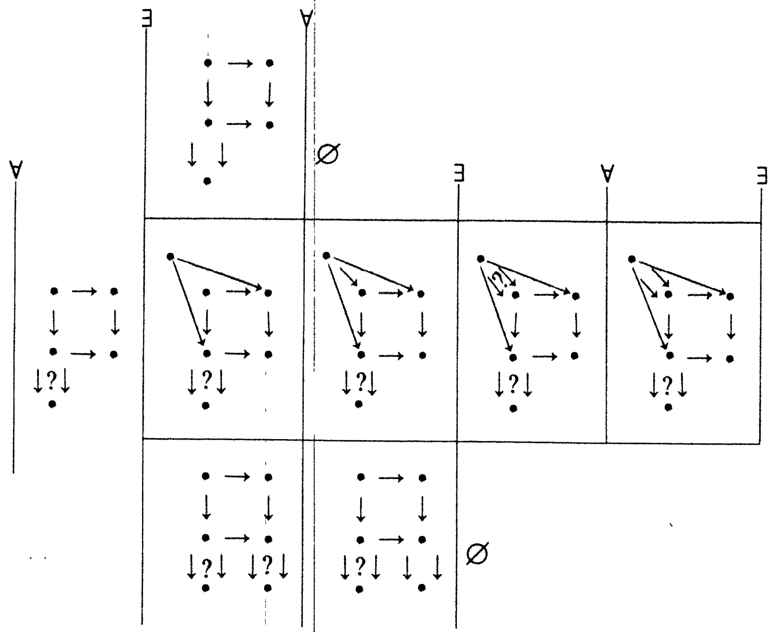

Fig. 11

Given a CG-tree T with root R define the complementary tree T' as that obtained by transposing ∀ and ∃. Then $D: R \to \mathbb{A}$ satisfies T' if and only if it does *not* satisfy T. Diagrammatic sentences are closed under the usual Boolean operators of negation, conjunction, and disjunction. If one excepts the source-target information in the root, then the Boolean operators are available for CG-trees. Note that the labels ∀ and ∃ serve both as quantifiers and as conjunctions and disjunctions. Over the years we have developed notations to avoid nonlinear trees. For example, if Fig. 9 denotes a pullback and $\twoheadrightarrow$ an epimorphism, then the property that pullbacks transfer epimorphisms is that shown in Fig. 10. If one does not use such notation, then we are forced to the nonlinear tree in Fig. 11.

2. The Theorem

Theorem. An elementary property on categories is invariant within equivalence types of categories if and only if it is a diagrammatic property.

Outline of Proof. Induction does now work for the easy direction. That is, if $F: \mathbb{A} \to \mathbb{B}$ is an equivalence of categories (use only that it is full, faithful with a representative image) for any CG-tree T with root R and diagram $D: R \to \mathbb{A}$ that satisfies T, then $R \to A \xrightarrow{F} \mathbb{B}$ also satisfies T. The diagrammatic notation successfully avoids the anomalies that result from the Frege notation.

For the other direction, we define a CI-graph as a C-graph together with a distinguished set of arrows, called "identity conditions." If G is a CI-graph, then $D: G \to \mathbb{A}$ is a diagram if besides respecting the commutativity conditions it carries the distinguished arrows into identity maps in $\mathbb{A}$. Just as above we define CIG-trees and what it means for a diagram from the root to satisfy a CIG-tree. Say that a graph is simple if each vertex appears as a source or target at most once.

Lemma. For any elementary property $P(A_1, \ldots, A_n, x_1, \ldots, x_m)$ there is a CIG-tree T with simple root R with $\langle a_1 \cdots a_m \rangle$ as arrows, $\{v_1, \ldots, v_n, sa_1, ta_1, \ldots, sa_n, ta_n\}$ as vertices, such that $D: R \to \mathbb{A}$ satisfies T if and only if

$$P(D(v_1), \ldots, D(v_n), D(a_1), \ldots, D(a_m))$$

is true in $\mathbb{A}$. In particular, for every elementary sentence S there is a CIG-tree T with empty root such that $\varnothing \to \mathbb{A}$ satisfies T if and only if $\mathbb{A}$ satisfies S.

Lemma. For every CIG-tree T with empty root there is a CG-tree T' with empty root such that for all skeletal categories $\mathbb{A}$, $\varnothing \to \mathbb{A}$ satisfies T if and only if it satisfies T'.

This is the difficult lemma. One proves by a cumbersome induction over all trees, empty-rooted or not, that for every CIG-tree T with root R there is a CIG-tree T' with root R such that $D: R \to \mathbb{A}$ satisfies T if and only if it satisfies T' for all skeletal $\mathbb{A}$, where T' is such that all identity conditions involve only arrows that appear in the root. Hence if R is empty then T' is a CG-tree. T' tends to be much fatter than T.

The lemmas yield the theorem: If S is a sentence invariant within equivalence types, let T be an empty-rooted CG-tree such that $\varnothing \to \mathbb{A}$ satisfies T if and only if $\mathbb{A}$ satisfies S for all *skeletal* $\mathbb{A}$. Since every category is equivalent to a skeletal category and $\varnothing \to \mathbb{A}$ satisfying T is invariant within equivalence types and, by assumption, so is S, then $\varnothing \to \mathbb{A}$ satisfies T if and only if $\mathbb{A}$ satisfies S for all $\mathbb{A}$, skeletal or not. (By using the Gödel completeness theorem one needs only that all *countable* categories are equivalent to skeletal categories, and hence can avoid using the axiom of choice.)

3. Back to Frege

Consider the Frege language on two sorts: "objects," $A, B, C, \ldots$; "maps," $x, y, z, \ldots$; and atomic predicates $(x = y)$, $(A = B)$, $(xy = z)$, $(A = \Box x)$, $(A = x\Box)$, where the last two are pronounced "A is the source (target) of x." We wish to characterize those sentences invariant within equivalence type.

We shall interpret the "restricted quantifiers,"

$$\forall_{A \xrightarrow{x} B}[\cdots] \quad \text{as} \quad \forall_x[(A = \Box x) \wedge (B = x\Box) \Rightarrow \cdots]$$

and

$$\exists_{A \xrightarrow{x} B}[\cdots] \quad \text{as} \quad \exists_x[(A = \Box x) \wedge (B = x\Box) \wedge \cdots].$$

Note that $\neg \forall_{A \xrightarrow{x} B}[\cdots]$ is equivalent with $\exists_{A \xrightarrow{x} B} \neg [\cdots]$. A sentence will be called a **Frege-diagrammatic** sentence if all quantified maps are so restricted and

(1) No map is quantified without its source and target having been previously quantified;

(2) The atomic predicates $(A = \Box x)$, $(B = \Box x)$ do not appear other than implicitly in the restricted quantifiers;

(3) The atomic predicate $(A = B)$ does not appear;

(4) If $(x = y)$ appears as an atomic predicate then the restricted quantifiers for x and y imply that $\Box x = \Box y$ and $x\Box = y\Box$;

(5) If $(xy = z)$ appears as an atomic predicate then the restricted quantifiers for x, y, and z imply that $\Box x = \Box z$, $x\Box = \Box y$, and $y\Box = z\Box$;

(6) If $x = 1_A$ appears as an atomic predicate then the restricted quantifier implies $A = \Box x$ and $A = x\Box$.

It is routine that for an empty-rooted CG-tree T there is a Frege-diagrammatic sentence S such that $\varnothing \to \mathbb{A}$ satisfies T if and only if $\mathbb{A}$ satisfies S. Conversely, we can find for any Frege-diagrammatic sentence such a CG-tree. Hence, an elementary sentence S is invariant within equivalence types if and only if there is a Frege-diagrammatic sentence S' such that the axioms of category theory imply $S \Leftrightarrow S'$. There can be no algorithm, incidentally, for deciding whether an arbitrary sentence is invariant within equivalence types. (For any word problem for monoids there is a sentence S true for all categories if and only if the given word problem is true. $S \vee \forall_{A,B}(A = B)$ is invariant within equivalence types if and only if S is true for all categories.)

Linear CG-trees correspond to prenex Frege-diagrammatic sentences, that is, all quantities in front. The sentence

$$\forall_{A,B}[(\exists_{A \xrightarrow{x} B}(x = x)) \vee (\exists_{B \xrightarrow{x} A}(x = x))]$$

says that a category is linearly connected. It cannot be put in prenex Frege-diagrammatic form.

AMS 02-16, 18-10

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF PENNSYLVANIA
PHILADELPHIA, PENNSYLVANIA