Pushouts in Algebraic Theories

6.UAP Final Report

Lawrence Wu

llwu@mit.edu

Supervised by: David I. Spivak

May 20, 2016

Contents

1 Introduction

Category theory has seen a resurgence in popularity in the 21st century as fields outside of mathematics have realized its seemingly fundamental nature in the structure of information. Recently, computer science has been a driver of this trend, as category theory has been embraced in academic communities such as those of Haskell [1] and formal methods [2], as well as incorporated in undergraduate computer science courses [3]. In this project, we study a category theoretic construction which arises in informatics. [1]

1.1 Algebraic Theories

Borrowing from the field of universal algebra, an *algebra* is a set equipped with a collection of operations. A *theory* is a collection of axioms stating which operations, if any, must exist and which equational laws, if any, must be satisfied by these operations. A *model* of a theory is then an algebra satisfying the axioms of the theory. This view of theories and models is analogous to the database concept of schemas and instances - the analogy has been made explicit in the open source SQL-compatible implementation of Functional Query Language by Patrick Schultz, David Spivak, and Ryan Wisnesky [6], to which the results of this project would be of direct interest.

In Bill Lawvere's 1963 PhD thesis [7], Lawvere formalizes the above idea in categorical language. An *algebraic theory* is defined to be a category with finite products, algebras of that theory are functors from the algebraic theory to **Set** preserving finite products, and an algebraic category has algebras as objects and their natural transformations as morphisms. In this project, we study *single sorted* algebraic theories, in which there is one generic object T, so the objects of the algebraic theory are $T^0, T^1, T^2, \ldots$ and the theory is generated by operations $T^n \to T^k$ along with any equational laws. For example, the algebraic theory

[1] We frequently use common category theoretic terminology, which is catalogued online at the nLab [4].

for groups is generated by letting T^1 be a group object, i.e. letting there be operations $m : T^2 \to T$, $e : T^0 \to T^1$, $inv : T^1 \to T^1$ and equational laws $m \circ (m \times \mathrm{id}_{T^1}) = m \circ (\mathrm{id}_{T^1} \times m)$, $m \circ (\mathrm{id}_{T^1} \times e) = p_1$, $m \circ (e \times \mathrm{id}_{T^1}) = p_2$, $m \circ (\mathrm{id}_{T^1} \times inv) \circ \Delta = e_{T^1}$, $m \circ (inv \times \mathrm{id}_{T^1}) \circ \Delta = e_{T^1}$, where p_1 and p_2 are the canonical projections, Δ is the diagonal map, and e_{T^1} is the composition of the counit $G \to 1$ with e. Then a functor from this theory to **Set** takes T^1 to the underlying set, m to the binary operation, e to the identity, and inv to the inverse.

1.2 Pushouts

A *pushout* is an example of a *universal construction* in category theory. Given $Y \xleftarrow{g} X \xrightarrow{f} Z$, a pushout is an object P along with arrows f' and g', such that the following square commutes, and for any Q that would make the following square commute, there is a unique *universal arrow* from P to Q.

$$\begin{array}{ccc} X & \xrightarrow[g]{} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ Z & \xrightarrow[g']{} & P \end{array}$$

Pushouts generalize the idea of *summing* algebras; in the context of databases, they arise when merging instances. For applications, we would like some guarantees on the behavior of pushouts; in particular, we would like pushouts to satisfy the following:

Property 1.1. *If f and g are monomorphisms, then f' and g' are monomorphisms.*

Here, *monomorphism* is analogous to *injective function* (we will see later that these are in fact the same) - we want no two elements to map to the same element, otherwise in applications we would have to handle nonsensical results such as $1 = 2$. We know that this property holds true in some, but not all algebraic categories, although a useful pattern has not been recognized; this project is dedicated to understanding pushouts in algebraic categories and in particular, when this property does or does not hold.

2 Existing Literature

Property 1.1 has been considered over all categories [8]; there is a class of categories called *adhesive categories* which satisfy the property. However, we find no existing criteria, even for algebraic categories, which are useful in practice [11]. We do, however, know of some cases of algebraic categories where the property does or does not hold. We survey these examples in the hopes that they will reveal a pattern, or that other cases can be reduced to these cases (we have found this to be somewhat fruitful for the **Set** case).

2.1 Non-Pathological Cases

From *Categories for the Working Mathematician* [9], we know that Property 1.1 holds in **Grp** and any abelian category (notably **Ab**) and also that the pushout $Y \bigsqcup_X Z$ in **Set** is the disjoint union of Y and Z identifying fx and gx for each $x \in X$, along with the inclusion maps. The following is also thus known:

Lemma 2.1. *Property 1.1 holds in* **Set**.

Proof. First, note that the monomorphisms of **Set** are exactly the injective functions. Indeed, if $f : X \to Y$ is injective then for all g_1, $g_2 : Z \to X$, we have for all $z \in Z$, $fg_1(z) = fg_2(z)$ implies $g_1(z) = g_2(z)$. Thus, $fg_1 = fg_2$ implies $g_1 = g_2$ so f is monic. Conversely, if f is monic, then $f(x) = f(x')$ implies $fg_1 = fg_2$ so $g_1 = g_2$ so $x = x'$, where g_1 is the mapping from a singleton to x and g_2 is the mapping from the singleton to x'. Now, in a set pushout:

$$\begin{array}{ccc} X & \xrightarrow[g]{} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ Z & \xrightarrow[g']{} & P \end{array}$$

if f and g are monic, then they are injective, so the pairs fx, gx are disjoint. Thus no elements are identified in f' or g', so f' and g' are injective and thus also monic. □

2.2 Pathological Cases

Property 1.1 does not hold in **CRing**, the algebraic category of commutative rings, by the following counterexample, quoted from [10]:

> Let k be a field, let $R = k[a,b]/(a^2, ab, b^2)$, $A = R[x]/(a - bx, x^2)$,
> $B = R[y]/(b - ay, y^2)$; then
>
> $$A \otimes_R B \cong R[x,y]/(a-bx, x^2, b-ay, y^2) \cong k[a,b,x,y]/(a^2, ab, b^2, a-bx, x^2, b-ay, y^2)$$
>
> but
>
> $$a = (a - bx) + (b - ay)x + (a - bx)xy + (x^2)by$$
> $$b = (b - ay) + (a - bx)y + (b - ay)xy + (y^2)ax$$
>
> so $(a, b) \subseteq \ker(R \to A \otimes_R B)$.

Propert 1.1 also fails in **CMon**, the algebraic category of commutative monoids, by the following counterexample, quoted from [11]:

> Take $A = \{u, v, 1, 0\}$ to be any commutative monoid with identity element 1 and null element 0. Define $B = A \sqcup \{b\}$, adjoining a new element b to A, and extending the multiplication on A by $bu = ub = v$ and $b \cdot 1 = 1 \cdot b = b$ and $bx = xb = 0$ for any $x \notin \{u, 1\}$. Similarly define $C = A \sqcup \{c\}$ with $cv = vc = u$ and $c \cdot 1 = 1 \cdot c = c$ and $cx = xc = 0$ for $x \notin \{v, 1\}$. Let $i_B : A \to B$ and $i_C : A \to C$ be the inclusion maps, and let P be the pushout of i_B and i_A with coproduct coprojections $j : B \to P$, $k : C \to P$. I claim $k \circ i_C : A \to P$ is not monic. The short version of the calculation is that in P we have

$$u = vc = buc = b \cdot 0 = 0$$

3 Approach

We build up from the most basic results. Restricting our attention to single-sorted algebraic theories allows us to reason more easily about the resulting algebraic categories.

3.1 Using The Underlying Set Functor

Any algebraic category **T-Alg** has an underlying set functor $U : \mathbf{T\text{-}Alg} \to \mathbf{Set}$ (in the case of single-sorted theories, taking $A^0, A^1, \ldots$ to A^1), which has a left adjoint, the free functor [12]. Since every map between sets from T^n is a product of the maps between sets from T^1, U is faithful. Note that $f : A \to B$ is a monomorphism iff the following is a pullback square:

$$\begin{array}{ccc} A & \xrightarrow[1_A]{} & A \\ {\scriptstyle 1_A}\downarrow & & \downarrow{\scriptstyle f} \\ A & \xrightarrow[f]{} & B \end{array}$$

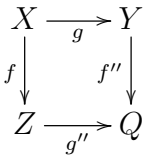
but pullbacks are limits, and right adjoints preserve limits, so right adjoints preserve monomorphisms, so U preserves monomorphisms. Also, if $U(f)$ is monic, then for all $g_1, g_2 \in \mathbf{T\text{-}Alg}$, $fg_1 = fg_2$ implies $U(f)U(g_1) = U(f)U(g_2)$ by functoriality, which implies $U(g_1) = U(g_2)$ by monicity, which implies $g_1 = g_2$ by faithfulness. Hence, U also *reflects* monomorphisms in addition to preserving them. We can now think of monomorphisms in algebraic category as follows:

Remark 3.1. *An arrow in an algebraic category is monic if and only if the underlying function between sets is injective.*

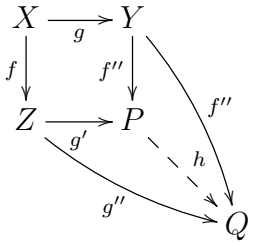
3.2 Adding Operations to the Theory

We noted, in Lemma 2.1, that Property 1.1 holds for **Set**, for which the algebraic theory has no operations and no equational laws. Since we have seen that equational laws potentially cause trouble (e.g. in **CMon** and **CRing**), we study what happens if we have any operations but no equational laws. Intuitively, such algebraic categories should not behave significantly differently from **Set**, so our approach relates Lemma 2.1, the underlying set functor, and the following fact, to extend Property 1.1 from **Set** to arbitrary algebraic theories without equational laws.

Lemma 3.1. *If there is Q such that the following square commutes with all arrows monic, then the pushout square of $Y \bigsqcup_X Z$ is also monic.*



Proof. The universal property gives us an arrow $h : P \to Q$:



Since $g'' = hg'$ is a monomorphism, g' is also a monomorphism, since

$$g'k_1 = g'k_2 \implies hg'k_1 = hg'k_2 \implies k_1 = k_2.$$

Similarly, f' is a monomorphism. □

Now, we are ready to prove the main result of this section.

Proposition 3.1. *Property 1.1 holds in any algebraic category with no equational laws.*

Proof. Let $\mathbf{T} = T^0, T^1, \ldots$ be an algebraic theory with no equational laws, and **T-Alg** be the category of algebras of $\mathbf{T}$. Suppose, in **T-Alg**, $Y \overset{g}{\leftarrow} X \overset{f}{\rightarrow} Z$ has f and g monic. Since f and g are natural transformations (of algebras, which are **Set**-valued functors), they commute with each operation $s : T^n \to T$ of the theory. Now consider the following pushout in **Set**:

$$\begin{array}{ccc} U(X) & \xrightarrow[U(g)]{} & U(Y) \\ {\scriptstyle U(f)}\downarrow & & \downarrow{\scriptstyle f'} \\ U(Z) & \xrightarrow[g']{} & Q^1 \end{array}$$

If we can find a square containing $Y \overset{g}{\leftarrow} X \overset{f}{\rightarrow} Z$ which commutes and is taken to the above square by U, then we are done, since Lemma 2.1 and the fact that U reflects monomorphisms would imply that this square satisfies the assumptions of Lemma 3.1. Let $f'' : Y \to Q$ be the natural transformation consisting of $(f')^n : Y^n \to Q^n := (Q^1)^n$ for each $n \in \mathbb{N}^0$ and $g'' : Z \to Q$ be similarly formed by products of g'. For these to exist, we need an algebra Q defined on underlying set Q^1 with each operation $s : T^n \to T$ defined on Q to commute with f'' and g''. If Q^1 is empty, then so are Y and Z, so the claim is trivially true. Otherwise, we can pick an arbitrary element r of Q^1. Note that for $q \in Q^n$, if $q = f''(y) = g''(z)$ for some $y \in Y^n, z \in Z^n$ then since Q^1 is a set pushout, there are $x_1, x_2 \in X^n$ such that $y = g(x_1), z = f(x_2)$, but since the natural transformation $f''g = g''f$ is composed of injective functions (Remark 3.1), we have $x := x_1 = x_2$, so

$$f''(s_Y(y)) = f''(s_Y(g(x))) = f''(g(s_X(x))) = g''(f(s_X(x))) = g''(s_Z(f(x))) = g''(s_Z(z)).$$

Thus we are allowed to define

$$s_Q(q) = \begin{cases} f''(s_Y(y)) & q = f''(y) \\ g''(s_Z(z)) & q = g''(z) \\ r & \text{neither of the above} \end{cases}$$

Now s clearly commutes with f'' and g'', as desired. $\square$

Note that the **CMon** counterexample shows how this proof breaks down when equational laws are introduced: if we try to apply the construction from our proof, then $s_Q(b, S_Q(u, c)) = s_Q(b, 0) = 0$ but $s_Q(s_Q(b, u), c) = s_Q(v, c) = u$, breaking the equational law for associativity! We suspect that products in equational laws break our construction by "mixing" codomains - indeed, we can extend our result to support this idea.

Proposition 3.2. *Property 1.1 holds in any algebraic category with no equational laws relating operations of arity greater than* 1.

Proof. We can rewrite the theory with all equational laws of the form $s = ab$, since the codomain of a product can only the the domain of an operation of arity greater than 1. Now construct Q as in the proof of Proposition 1.1. If $q = f''(y)$ then $b_Q(q) = f''(b_Y(y))$ so $a_Q(b_Q(q)) = f''(a_Y(b_Y(y))) = s_Q(q)$. Similarly, if $q = g''(z)$ then $b_Q(q) = g''(b_Z(z))$ so $a_Q(b_Q(q)) = g''(a_Z(b_Z(z))) = s_Q(q)$. By definition of Q^1, one of the previous assumptions on q must hold, so the equational laws are satisfied by this construction. $\square$

Proposition 3.1 implies that Property 1.1 holds in the categories of pointed sets (there is one operation $T^0 \to T^1$) and magmas (there is one operation $T^2 \to T^1$).

Example: Magmas
Suppose we have

$$X = \{1, 2\}$$

$$s_X(1,1) = 1, s_X(1,2) = 1, s_X(2,1) = 1, s_X(2,2) = 2$$

$$Y = \{1, 2, 3\}$$

$$s_Y(1,1) = 1, s_Y(1,2) = 1, s_Y(2,1) = 1, s_Y(2,2) = 2,$$

$$s_Y(1,3) = 2, s_Y(2,3) = 2, s_Y(3,1) = 2, s_Y(3,2) = 2, s_Y(3,3) = 3$$

$$Z = \{1, 2, 4\}$$

$$s_Z(1,1) = 1, s_Z(1,2) = 1, s_Z(2,1) = 1, s_Z(2,2) = 2$$

$$s_Z(1,4) = 4, s_Z(2,4) = 4, s_Z(4,1) = 4, s_Z(4,2) = 4, s_Z(4,4) = 4$$

where every map is the inclusion. Then our construction gives (with $r = 1$)

$$Q = \{1, 2, 3, 4\}$$

$$s_Q(1,1) = 1, s_Q(1,2) = 1, s_Q(2,1) = 1, s_Q(2,2) = 2$$

$$s_Q(1,3) = 2, s_Q(2,3) = 2, s_Q(3,1) = 2, s_Q(3,2) = 2, s_Q(3,3) = 3$$

$$s_Q(1,4) = 4, s_Q(2,4) = 4, s_Q(4,1) = 4, s_Q(4,2) = 4, s_Q(4,4) = 4$$

$$s_Q(3,4) = 1, s_Q(4,3) = 1$$

4 Conclusion

We were able to prove a result which we did not know previously (Proposition 3.1). This project proved instructive about algebraic theories, and category theory in general. Existing techniques surveyed here and techniques developed here are of potential use in attacking the

problem further.

4.1 Further Work

We currently have not explored an easy way to generate and reason about counterexamples - these could be studied in the future. Conditions could also be greatly refined - Lemma 3.1 gives a straightforward, but not obviously useful in general, condition for Property 1.1 to hold. While Lemma 3.1 does not seem computationally applicable, Proposition 3.1 showed that it was mathematically useful for proving Property 1.1 on an entire class of algebraic categories. Having such a guarantee is itself reassuring in applications, and refining the method to achieve further guarantees would be desirable. However, examples such as **Ab** show that we will need a more flexible construction than that of Proposition 3.1, which is potentially grounds for new approaches.

4.2 Acknowledgements

The author would like to thank David Spivak for his input and supervision of this work.

References

[1] https://www.haskell.org/tutorial/monads.html

[2] https://homotopytypetheory.org/

[3] http://www.cs.cornell.edu/courses/cs6117/2014Fa/

[4] https://ncatlab.org/nlab/show/HomePage

[5] J. Adámek; J. Rosický; E. M. Vitale (18 November 2010). *Algebraic Theories: A Categorical Introduction to General Algebra*. Cambridge University Press. ISBN 978-1-139-49188-4.

[6] http://categoricaldata.net/fql.html

[7] F. W. Lawvere, *Functorial Semantics of Algebraic Theories*, Ph.D. thesis, Columbia University, 1963.

[8] Martin Brandenburg, *When is the pushout of a monic also monic?*, URL (version: 2012-06-06): http://math.stackexchange.com/q/154591

[9] Lane, Saunders Mac. *Categories for the Working Mathematician*. New York: Springer-Verlag, 1971. Print.

[10] Zhen Lin, *Tensor product of injective ring homomorphisms*, URL (version: 2014-09-02): http://math.stackexchange.com/q/916443

[11] Todd Trimble, *Monomorphisms in operad algebras*, URL (version: 2016-03-05): http://mathoverflow.net/q/232864

[12] https://ncatlab.org/nlab/show/Lawvere+theory#FreeAlgebras