Formalizing the Algebraic Small Object Argument in UniMath

Dennis Hilhorst
Department of Mathematics, Utrecht University, The Netherlands

Paige Randall North
Department of Information and Computing Sciences, Department of Mathematics, Utrecht University, The Netherlands

Abstract

Quillen model category theory forms the cornerstone of modern homotopy theory, and thus the semantics of (and justification for the name of) *homotopy type theory / univalent foundations* (HoTT/UF). One of the main tools of Quillen model category theory is the *small object argument*. Indeed, the particular model categories that can interpret HoTT/UF are usually constructed using the small object argument.

In this article, we formalize the *algebraic* small object argument, a modern categorical version of the small object argument originally due to Garner, in the Coq `UniMath` library. This constitutes a first step in building up the tools required to formalize – in a system based on HoTT/UF – the semantics of HoTT/UF in particular model categories: for instance, Voevodsky's original interpretation into simplicial sets.

More specifically, in this work, we rephrase and formalize Garner's original formulation of the algebraic small object argument. We fill in details of Garner's construction and redefine parts of the construction to be more direct and fit for formalization. We rephrase the theory in more modern language, using constructions like *displayed categories* and a modern, less strict notion of *monoidal categories*. We point out the interaction between the theory and the foundations, and motivate the use of the algebraic small object argument in lieu of Quillen's original small object argument from a constructivist standpoint.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases formalization of mathematics, univalent foundations, model category theory, algebraic small object argument, coq, unimath

Digital Object Identifier 10.4230/LIPIcs.ITP.2024.20

Supplementary Material *Software (Source code)*: `https://github.com/UniMath/UniMath/tree/6605a4a/UniMath/ModelCategories`
archived at `swh:1:rev:6605a4a22a7dcda5d27025b9c75ae2f905338b6f`
InteractiveResource (Interactive version of source code): `https://math.asoa.dennishilhorst.nl/`
Other (Contribution PR): `https://github.com/UniMath/UniMath/pull/1822`
Other (Optimization PR): `https://github.com/UniMath/UniMath/pull/1858`

Funding *Paige Randall North*: This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0334.

1 Introduction

The tools of model category theory underpin the semantics of homotopy type theory and univalent foundations (HoTT/UF) (starting with [7, 19]). This work is the first step in building up this toolkit within HoTT/UF, with the ultimate goal of formalizing and verifying the semantics of HoTT/UF within HoTT/UF – or more specifically, within the Coq library `UniMath`. We focus on formalizing one major tool in the envisioned kit: the algebraic small object argument. This is the main tool for constructing particular model categories, and it (or versions) is used to model HoTT/UF in particular categories [19, 5, 8, 11, 10, 6].

Model category theory and the algebraic small object argument

Model category theory, first introduced by Quillen [25], forms the foundation of modern homotopy theory. It provides a language and tools for this branch of mathematics, expanding the use of methods originally developed for the study of topological spaces to other domains: e.g. (higher) category theory [18, 17, 26], derived algebraic geometry [21], motivic homotopy theory [23], and now type theory as mentioned above.

A model category consists of two interacting *weak factorization systems* (WFSs) on a category, and, in specific examples, these are usually constructed via the *small object argument* [25] or variations. In this article, we study and formalize *algebraic/natural* weak factorization systems (NWFSs) [14] and the *algebraic* small object argument [13, 12], categorical improvements to the original versions that are well-suited for formalization. (Though *algebraic weak factorization system* is the more modern terminology, we will adhere to the vocabulary from [13] and refer to them as *natural weak factorization systems*.)

WFSs, while important for the theory of model categories, are lacking from a categorical or constructivist viewpoint. The definition and the problems created will be explicated in Section 3, but in summary a WFS consists of some structure A together with the property that some other structure B merely exists. NWFSs solve the constructivist problems that this creates by including both A and B explicitly, as structure, not property. In addition, to take advantage of machinery from category theory, B is not only given explicitly as structure, but as arising from (co)algebras of a (co)monad. Though the definition of NWFS is, *a priori*, more restrictive than that of WFS, most WFSs of interest to us (e.g. the ones in [19, 5, 8, 11, 10, 6]) are actually NWFSs. Additionally, NWFSs are an important tool in recent efforts towards producing constructive models of HoTT/UF [11, 10].

Quillen's *small object argument* (SOA) [25] generates a WFS from a sufficiently well-behaved subclass of maps of a category. Garner introduced a variant, the *algebraic small object argument* (ASOA) [13] which produces NWFSs and which takes advantage of the fact that NWFSs are defined in terms of (co)monads in order to produce a cleaner construction. The NWFSs of [19, 5, 8, 11, 10, 6] are generated by the ASOA or variations.

Coq and UniMath

Coq in conjunction with the `UniMath` library (henceforth just `UniMath`) is a formalization framework for HoTT/UF. It adds insights from homotopy theory to type theory to produce a foundation of mathematics that is well-suited for the formalization of mathematics closely related to homotopy theory, especially the category theory that we are using here.

We only make light use of the additional assumptions that `UniMath` adds to Coq: we use functional extensionality and the homotopy levels of propositions and sets, as well as propositional truncation for the classical theory in Section 3 (though *not* in the actual ASOA construction in Section 5), but nothing else (including univalence). We do however take significant advantage of the technology developed in `UniMath` for formalizing category theory. In particular, we use the machinery of *displayed categories* [2] and the extensive formalization of *monoidal categories* [28]. Furthermore, we do expect the univalence axiom to become important and useful for the further development of model category theory in `UniMath`.

That being said, some of the high level machinery employed in [13] is not available in the `UniMath` library. For example, much of the theory of (co)ends and the two-fold monoidal categories employed there are not yet available in `UniMath`. Instead, we make more direct arguments, making for a more concrete and detailed description of the construction.

Contributions and related work

We formalize [13, Proposition 4.22] with the "smallness requirement" (in the language of [13]) that every object is finitely presentable (see Theorem 50), and its prerequisites. Garner also proves Proposition 4.22 for other smallness requirements. What we have formalized constitutes the most important parts of the algebraic small object argument construction in the most general setting with the current available theory, and is still applicable to situations of interest. See Remark 39 for technical details.

We have modified the proofs to be feasible in `UniMath`. We use the machinery of displayed categories [2] to construct various categories used in [13]; this is necessary to talk about functors commuting strictly (i.e., up to definitional equality), as is done in [13]. Where [13] uses strict monoidal categories, we use the weaker monoidal categories of [28], to make our constructions more widely applicable. We give the construction of the free monoid (see Section 5.4) and the proofs of [13, Proposition 4.18] (Section 5.3.1) and [13, Propositions 4.19 and 4.22] (Section 5.5) more directly. Furthermore, the formalization provides a more detailed and streamlined account for the proof of [13, Proposition 4.17].

Our formalization has been accepted into the `UniMath` library in commit `6605a4a`.

This paper is based on a master's thesis by the first-named author [15]. This provides an expanded account of our formalization with more details and diagrams.

Outline and preliminary remarks

We first introduce the reader to relevant aspects of `UniMath` in Section 2. We then introduce WFSs and the SOA in Section 3, pointing out constructive issues. We introduce NWFSs in Section 4, showing that they fix the issues encountered in WFSs. Finally, we go over the algebraic small object argument in Section 5, going into some detail on our modifications.

Composition of morphisms in this paper is written in *diagrammatic order*, adopting the conventions in the `UniMath` library (so the composite of $f : X \to Y$ and $g : Y \to Z$ is written $f \cdot g$). Throughout, we assume $\mathcal{C}$ to be a cocomplete category.

2 Preliminary theory in HoTT/UF

2.1 Homotopy levels

There is a hierarchy of *homotopy n-types* in `UniMath`, indexed by $n : \mathbb{N}$. We use 1-types, called *mere propositions*, and 2-types, called *sets*. A type P is a *mere proposition* if any two points are equal (meaning it may be empty): i.e., if there is a term of $\prod_{x,y:P}(x =_P y)$. We denote the type of mere propositions by `hProp` [27, `hProp`]. A type is a *set* if all of its identity types are propositions. We denote the type of sets by `hSet` [27, `hSet`].

Sometimes, instead of needing a term of a type A, it is sufficient (or perhaps even necessary!) to only know of the *mere existence* of a term of A. That is to say, we want a propositional type witnessing only that "a term of type A exists", ignoring what this term is exactly or how it is constructed. This idea is captured in the *propositional truncation of A*, denoted $\|A\|$. `UniMath` defines $\|A\|$ through an "impredicative encoding" [27, `ishinh_UU`]

$$\|A\| := \prod_{P:\texttt{hProp}} ((A \to P) \to P).$$

▶ Remark 1. It is important to note that we can *not* (in general) obtain a term of type A given a term of type $\|A\|$. We are effectively *losing* information when truncating a type. This is the root cause for the constructive issues arising in the classical theory of WFSs.

2.2 Category theory

In set-based mathematics, a category consists of a *set* O of objects, and for each $X, Y \in O$ a *set* of morphisms $\hom(X, Y)$. In HoTT, one might naively define a category to consist of a *type* of objects and *types* of morphisms. We call this a *precategory.* Though allowing for basic constructions like limits [27, `Limits.v`] or colimits [27, `Colimits.v`], these are not sufficient for our purposes.

If we restrict the types $\hom(X, Y)$ to be sets, we get the notion of a *category.* In the rest of this paper, we will only be considering categories.

We can add more restrictions to this notion of category to produce *univalent categories* [1] or *setcategories* [4]. Interestingly enough, we *never* need to assume either of these restrictions for our categories, which makes our construction applicable to both.

2.3 Displayed categories

It is common practice to construct a new category $\mathcal{D}$ out of a category $\mathcal{C}$ by adding data or properties to the objects and morphisms, often expressed in terms of a forgetful functor $F : \mathcal{D} \longrightarrow \mathcal{C}$. Instead of mapping the objects and morphisms of $\mathcal{D}$ to those of $\mathcal{C}$, it is useful to index them as families of objects and morphisms "lying over" those of $\mathcal{C}$. This idea is captured in the notion of *displayed categories*, analogous to dependent type families in HoTT [2].

▶ **Definition 2** ([27, `disp_cat`]). *A* displayed category $\mathcal{D}$ *over* $\mathcal{C}$ *consists of the following:*

(i) *For each object $X : \mathcal{C}$, a type $\mathcal{D}_X$ of "objects over X";*

(ii) *For each morphism $f : X \to Y$ with $X, Y : \mathcal{C}$, and for each displayed object $\overline{X} : \mathcal{D}_X$ and $\overline{Y} : \mathcal{D}_Y$ a set of "morphisms from $\overline{X}$ to $\overline{Y}$ over f", denoted by $\overline{X} \to_f \overline{Y}$;*

(iii) *For each object $X : \mathcal{C}$ and each $\overline{X} : \mathcal{D}_X$, a displayed identity morphism $1_{\overline{X}} : \overline{X} \to_{\mathrm{id}_X} \overline{X}$;*

(iv) *For all $X, Y, Z : \mathcal{C}$, $\overline{X} : \mathcal{D}_X$, $\overline{Y} : \mathcal{D}_Y$, $\overline{Z} : \mathcal{D}_Z$ and $f : X \to Y$, $g : Y \to Z$, a composition*

$$(-) \cdot (-) : \left(\overline{X} \to_f \overline{Y}\right) \times \left(\overline{Y} \to_g \overline{Z}\right) \to \left(\overline{X} \to_{f \cdot g} \overline{Z}\right);$$

satisfying displayed versions of associativity and identity axioms.

One can easily construct an "actual" category from a displayed category, analogous to the construction of dependent pair types from type families in HoTT. `UniMath` calls the constructed category $\mathcal{D}^{\mathrm{tot}}$: the *total category of* $\mathcal{D}$. It provides *definitional* information on the relation of $\mathcal{D}^{\mathrm{tot}}$ to $\mathcal{C}$, whereas a forgetful functor would provide *propositional* information.

In the rest of this section, $\mathcal{D}$ will always denote a displayed category over $\mathcal{C}$.

▶ **Definition 3** ([27, `total_category`]). *The* total category $\mathcal{D}^{\mathrm{tot}}$ of $\mathcal{D}$ *is defined to have*

- **Objects***:* $\sum_{X:\mathcal{C}} \mathcal{D}_X$
- **Morphisms** *from the dependent pairs $(X, \overline{X})$ to $(Y, \overline{Y})$: the set $\sum_{f:X\to Y} \overline{X} \to_f \overline{Y}$*

with the natural unit and composition.

▶ Remark 4. For readability, we will mostly refer to total categories *without* their explicit notation. That is to say, we may use $\mathcal{D}$ to denote $\mathcal{D}^{\mathrm{tot}}$ for a displayed category $\mathcal{D}$.

▶ **Example 5** (`arrow`). The *arrow category* $\mathcal{C}^{\mathbf{2}}$ of $\mathcal{C}$ is the total category of a displayed category over $\mathcal{C} \times \mathcal{C}$ with

- **Displayed objects** over $(X, Y) : \mathcal{C} \times \mathcal{C}$ as the type of arrows $X \to Y$.
- **Displayed morphisms** between displayed objects $f : X \to Y$ and $g : A \to B$ over a morphism $(h, k) : (X, Y) \to (A, B)$: the proposition $f \cdot k =_{(X \to B)} h \cdot g$.

It is the functor category from the poset $\mathbf{2} := \{0, 1\}$ to $\mathcal{C}$.

▶ **Example 6** (`three`)**.** The *three category* $\mathcal{C}^{\mathbf{3}}$ of $\mathcal{C}$ is the total category of a displayed category over $\mathcal{C}^{\mathbf{2}}$ with

- **Displayed objects** over $f : X \to Y : \mathcal{C}^{\mathbf{2}}$:

$$\sum_{(E_f:\mathcal{C})} \sum_{(f_{01}:X\to E_f)} \sum_{(f_{12}:E_f\to Y)} f_{01} \cdot f_{12} =_{(X\to Y)} f.$$

- **Displayed morphisms** between displayed objects (E_f, f_{01}, f_{12}) and $(E_{f'}, f'{}_{01}, f'_{12})$ over a morphism (g_{00}, g_{22}):

$$\sum_{g_{11}:E_f\to E_{f'}} \left(f_{01} \cdot g_{11} =_{(X\to E_{f'})} g_{00} \cdot f'_{01}\right) \times \left(f_{12} \cdot g_{22} =_{(E_f\to Y')} g_{11} \cdot f'_{12}\right).$$

It is the functor category from the poset $\mathbf{3} := \{0, 1, 2\}$ to $\mathcal{C}$.

▶ Remark 7. Though both of the previous examples are equivalent to certain functor categories from posets, defining them in terms of displayed categories provides definitional equalities that are much simpler to reason with in formalization. From a classical point of view, one may consider them to be functor categories.

`UniMath` defines displayed variants of functors and natural transformations, in such a way that they "lift" to functors and natural transformations on the total categories. We are mostly interested in one kind of displayed functor: *sections of displayed categories.* Given a category $\mathcal{C}$ and a displayed category $\mathcal{D}$ over $\mathcal{C}$ there is a projection functor $\pi_1^{\mathcal{D}} : \mathcal{D}^{\text{tot}} \longrightarrow \mathcal{C}$ projecting a pair $(X, \overline{X})$ down to X. Morally, a section is a *strict* right inverse to $\pi_1^{\mathcal{D}}$.

▶ **Definition 8** ([27, `section_disp`])**.** *A* section *from $\mathcal{C}$ to $\mathcal{D}$ consists of a dependent function of objects $F : \prod_{X:\mathcal{C}} \mathcal{D}_X$ and a corresponding dependent function, also denoted F, of type $\prod_{f:X\to Y} F(X) \to_f F(Y)$, such that $F(\mathrm{id}_X) =_{F(X)\to F(X)} 1_{F(X)}$ and $F(f \cdot g) =_{F(X)\to F(Z)} F(f) \cdot F(g)$ for morphisms $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{C}$. Such a section lifts to a functor $\mathcal{C} \longrightarrow \mathcal{D}^{tot}$.*

▶ Remark 9. For any section $F : \mathcal{C} \longrightarrow \mathcal{D}$ and any $X : \mathcal{C}$, the composite $(F \cdot \pi_1^{\mathcal{D}})(X)$ is in fact *definitionally equal* to X. There is no way to specify this definitional equality using a forgetful functor. The definitional equality is much more convenient to reason with, greatly simplifying the formalization process. Additionally, it is necessary to faithfully capture the classical theory that we are formalizing: see Remark 27.

We define a notion of natural transformation between sections, again to capture *strict* commutation over $\mathcal{C}$ (one corresponding to whiskering a natural transformation with $\pi_1^{\mathcal{D}}$).

▶ **Definition 10** (`section_nat_trans_disp`)**.** *Let F, F' be sections of the displayed category $\mathcal{D}$ over $\mathcal{C}$. A* natural transformation of sections *from F to F' is a family of displayed morphisms*

$$\prod_{X:\mathcal{C}} F(X) \to_{\mathrm{id}_X} F'(X),$$

making the appropriate diagrams commute.

3 Weak factorization systems

Weak factorization systems consist of two subclasses of morphisms (`morphism_class`) of a category $\mathcal{C}$, related through lifting properties, as well as a factorization of all morphisms. We define the lifting properties in terms of *lifting problems* and *fillers*. We use propositional truncation in both the lifting properties and the factorization.

▶ **Definition 11** (`filler`). *For morphisms f, g in $\mathcal{C}$, an (f, g)-*lifting problem *is a morphism $f \to g$ in $\mathcal{C}^2$. More precisely, it is a commutative square as in the left-hand diagram below.*

$$\begin{tikzcd} X \arrow[r] \arrow[d, "f"'] & A \arrow[d, "g"] \\ Y \arrow[r] & B \end{tikzcd} \rightsquigarrow \begin{tikzcd} X \arrow[r] \arrow[d, "f"'] & A \arrow[d, "g"] \\ Y \arrow[r] \arrow[ur, dashed, "l"'] & B \end{tikzcd}$$

We call a diagonal map $l : Y \to A$ that makes the whole diagram commute a filler.

▶ **Definition 12** (`lp`). *Given morphisms f, g in $\mathcal{C}$, we say that that (f, g) has the* lifting property *if there* merely exists *a filler for every (f, g)-lifting problem. That is,*

$$\texttt{lp} := \prod_{f:X\to Y} \prod_{g:A\to B} \prod_{x:f\to g} \left\| \sum_{l:Y\to A} l \text{ is a filler for } x \right\|.$$

▶ Remark 13. We use propositional truncation here so that `lp` is a proposition, as we want to use it to define subclasses of morphisms: see Remark 17 below. Still, we are able to show interesting properties using the recursion principle of the propositional truncation (`WFS.v`).

▶ **Definition 14** (`rlp`, `llp`). *We say that g has the* right lifting property *with respect to a subclass of morphisms $\mathcal{L}$ if (f, g) has the lifting property for all $f \in \mathcal{L}$. We denote the class of all such g as $\mathcal{L}^{\square}$. Dually, we say that f has the* left lifting property *with respect to a class $\mathcal{R}$ if (f, g) has the lifting property for all $g \in \mathcal{R}$. We denote the class of all such f by ${}^{\square}\mathcal{R}$.*

▶ **Definition 15** (`wfs_fact_ax`). *A pair of subclasses of morphisms $(\mathcal{L}, \mathcal{R})$* factors *$\mathcal{C}$ if for any $f : X \to Y$ there* merely exists *an object $E_f : \mathcal{C}$ and morphisms $\lambda_f : X \to E_f$ in $\mathcal{L}$ and $\rho_f : E_f \to Y$ in $\mathcal{R}$ such that $f = \lambda_f \cdot \rho_f$.*

▶ **Definition 16** (`wfs`). *A weak factorization system (WFS) is an ordered pair $(\mathcal{L}, \mathcal{R})$ of subclasses of morphisms in $\mathcal{C}$ that factors $\mathcal{C}$ and satisfies*

$$\mathcal{L} = {}^{\square}\mathcal{R} \quad \textit{and} \quad \mathcal{R} = \mathcal{L}^{\square}.$$

▶ Remark 17. The definition of a WFS shows why we need the propositional truncation in the lifting property: the equalities $\mathcal{L} = {}^{\square}\mathcal{R}$ and $\mathcal{R} = \mathcal{L}^{\square}$ would otherwise be ill-typed. Without the propositional truncation, corresponding notions of ${}^{\square}\mathcal{R}$ or $\mathcal{L}^{\square}$ would not simply be subclasses of morphisms, but rather subclasses of morphisms *with extra data*, containing information about the fillers in any appropriate lifting problem.

In any WFS $(\mathcal{L}, \mathcal{R})$, $\mathcal{L}$ is *left saturated* and $\mathcal{R}$ is *right saturated* [22, Prop 14.1.8]. Left saturation tells us that $\mathcal{L}$ contains all isomorphisms (`wfs_L_contains_isos`) and is closed under retracts (`wfs_L_retract`), pushouts (`wfs_closed_pushouts`), transfinite composition and coproducts (`wfs_closed_coproducts`). Right saturation is defined dually. Constructive issues arise in the last two closure properties. To illustrate, consider the following lemma.

▶ **Lemma 18** (`wfs_closed_coproducts`). *Assume the axiom of choice. A WFS $(\mathcal{L}, \mathcal{R})$ is closed under coproducts. That is to say: for a set I and a family of maps $\{ f_i : X_i \to Y_i \}_{i:I}$ such that $f_i \in \mathcal{L}$ for all $i : I$, the coproduct $f := \coprod_{i:I} f_i$ is also in $\mathcal{L}$.*

Proof. Consider a $g \in \mathcal{R}$ and an (f, g)-lifting problem. By the universal property of the coproduct, this is equivalent to a (f_i, g)-lifting problem for each $i : I$. We obtain the *mere existence* of fillers $l_i : Y_i \to A$ through the lifting properties of the f_i.

$$\begin{tikzcd}
\coprod_{i:I} X_i \arrow[r] \arrow[d, "\coprod_{i:I} f_i"'] & A \arrow[d, "g"] & & X_i \arrow[r] \arrow[d, "f_i"'] & A \arrow[d, "g"] \\
\coprod_{i:I} Y_i \arrow[r] & B \arrow[rr, phantom, "\rightsquigarrow"] & & Y_i \arrow[r] \arrow[ur, dashed, "l_i"] & B
\end{tikzcd}$$

Using the axiom of choice, we infer the *mere existence* of a morphism $\coprod_{i:I} l_i : \coprod_{i:I} Y_i \to A$ as a filler for the original lifting problem [24]. ◀

▶ Remark 19 (Constructive issues). Why do we need the axiom of choice in this proof? To put it shortly: because we lose information through the propositional truncation. We know of the *mere existence* of a lift in every individual diagram, but need to put all the lifts together to infer the mere existence of a lift in a "combined" diagram. We effectively want to construct a function

$$\left(\prod_{i:I}\left\| \sum_{l_i:Y_i\to A} f_i\cdot l_i = h \times l_i\cdot g = k\right\|\right) \to \left\| \sum_{l:\prod_{i:I} Y_i\to A} \prod_{i:I} f_i\cdot l(i) = h \times l(i)\cdot g\right\|.$$

This is precisely the statement of the axiom of choice in `UniMath`, which says that for any set X and any $L : X \to \texttt{hSet}$, and any $P : \prod_{x:X} L(x) \to \texttt{hProp}$, we have

$$\left(\prod_{x:X}\left\| \sum_{l_x:L(x)} P(x, l_x)\right\|\right) \to \left\| \sum_{l:\prod_{x:X} L(x)} \prod_{x:X} P(x, l(x))\right\|.$$

Thus, we are only able to show that the left class of a WFS is closed under coproducts by assuming the axiom of choice.

As described in Remark 13 and Remark 17, we cannot drop the propositional truncation in the definition of a WFS to fix this issue. It is resolved in the theory of NWFSs however, where the added algebraic structure provides a canonical choice function in analogous lifting problems, see Remark 33 and Lemma 34.

3.1 The small object argument

In this section, we briefly describe the SOA [25], following the account in [16]. We have not formalized the SOA; this section is intended to motivate and build intuition for the ASOA.

The SOA allows us to construct WFSs given a sufficiently well-behaved subclass of morphisms. The constructed WFS is related to the generating class J through the lifting property itself: its right class will be $J^{\square}$, and its left class will be ${}^{\square}(J^{\square})$.

For the rest of this paper, we assume J to be a subclass of morphisms in our category $\mathcal{C}$.

▶ **Definition 20.** *A* relative J-cell complex *is a transfinite composition of pushouts of morphisms in J. We denote this class by J-cell.*

▶ **Example 21.** The main motivating examples are in the categories of topological spaces (**TOP**) and simplicial sets (**SSET**). In both cases, J is the class of boundary inclusions (where $S^{-1} := \emptyset$)

$$J := \left\{\, j_n : S^n \hookrightarrow D^{n+1} \;\middle|\; n = -1, 0, 1, \ldots \right\}.$$

We think of pushing out one map of J along a function $f : S^n \to X$ (the *attaching map*) as producing a relative cell complex $X \to X \sqcup_{S^n} D^{n+1}$: that is, the inclusion of X into X with a copy of D^{n+1} (a *cell*) "glued" to it along f. Then we think of a relative J-cell complex as the inclusion of a space X into X with many cells attached. In topological spaces, these are called *relative CW-complexes*. When $X := \emptyset$ they are called *CW-complexes*.

▶ Remark 22. Assuming the axiom of choice, the class J-cell is a subclass of ${}^{\square}\big(J^{\square}\big)$ since then ${}^{\square}\big(J^{\square}\big)$ is closed under pushouts and transfinite compositions.

Let us briefly go over the SOA. We omit an explicit definition of the "smallness" giving rise to the name *small object argument*, but we will indicate when we use it.

▶ **Theorem 23** (Small object argument (SOA)). *Suppose the domains of all the maps in J are "small" relative to J-cell. Then there is a factorization $f \mapsto (\lambda_f, \rho_f)$ on $\mathcal{C}$ such that, for all morphisms f in $\mathcal{C}$, λ_f is in J-cell and ρ_f is in $J^{\square}$.*

Proof sketch. Let $f : X \to Y$ be a morphism in $\mathcal{C}$. We inductively construct factorizations

$$X \xrightarrow{\lambda_f^\alpha} E_f^\alpha \xrightarrow{\rho_f^\alpha} Y$$

of f, for ordinals α. First, we set $\lambda_f^0 = \mathrm{id}_X$ and $\rho_f^0 = f$. Since the composition of 0 morphisms is an instance of transfinite composition, id_X is in J-cell. However, f is not necessarily in $J^{\square}$; we continuously "improve" this factorization in the inductive step until some ρ_f^i is in $J^{\square}$.

Consider the factorization $f \mapsto (\lambda_f^\alpha, \rho_f^\alpha)$ corresponding to step α. Let S^α be the set of all (g, ρ_f^α)-lifting problems with g ranging over J. For any lifting problem $x : S^\alpha$, denote by $g_x : A_x \to B_x$ the corresponding map in J. We define $E_f^{\alpha+1}$ and $\rho_f^{\alpha+1}$ through the pushout on the left-hand side of the following diagram. The right-hand side shows the first step in the transfinite sequence.

$$\begin{tikzcd}
\bigsqcup_{x\in S^\alpha} A_x \arrow[r] \arrow[d, "\bigsqcup_{x\in S^\alpha} g_x"'] & E_f^\alpha \arrow[d, "s_\alpha"] \arrow[rd, bend left, "\rho_f^\alpha"] & & & \bigsqcup_{x\in S^0} A_x \arrow[r] \arrow[d, "\bigsqcup_{x\in S^0} g_x"'] & X \arrow[d, "\lambda_f^1"] \arrow[rd, bend left, "f"] & \\
\bigsqcup_{x\in S^\alpha} B_x \arrow[r] \arrow[rr, bend right] & E_f^{\alpha+1} \arrow[ul, phantom, "\ulcorner", very near start] \arrow[r, dotted, "\rho_f^{\alpha+1}"] & Y \arrow[r, phantom, "\overset{\alpha=0}{\rightsquigarrow}"] & & \bigsqcup_{x\in S^0} B_x \arrow[r] \arrow[rr, bend right] & E_f^1 \arrow[ul, phantom, "\ulcorner", very near start] \arrow[r, dotted, "\rho_f^1"] & Y
\end{tikzcd} \tag{1}$$

We set $\lambda_f^{\alpha+1} := \lambda_f^\alpha \cdot s_\alpha$, which is in J-cell by construction.

Note that the inductive process simply repeats the first step, meaning that

$$\rho_f^\alpha := \rho^1_{\rho^1_{\cdot^{\cdot^{\cdot}}\rho^1_f}} \qquad (\alpha \text{ times})$$

This defines the (successor ordinal part of the) transfinite construction of the small object argument. One can show that the smallness of the domains in J means that there is some ρ_f^α which is in $J^{\square}$. Very roughly, the cells being attached to the domain of f at each step are solutions to lifting problems between J and f; the smallness guarantees that at some point, solutions to all possible lifting problems have been added. ◀

▶ Remark 24 (Constructive issues). Besides the use of the axiom of choice mentioned in Remark 22, we note some other constructive and categorical issues in the argument. In the categories of topological spaces and simplicial sets, the construction boils down to the following: at every step in the transfinite construction, we glue on cells for every possible lifting problem. This means that at every step, we glue duplicate cells, as we can glue all the cells that we have glued before (in addition to new ones). Thus, the construction never converges; we simply just *stop* whenever we have gone far enough, dictated by the smallness assumption on J. This again implies some sort of choice, and may introduce massive ordinal sequences (which pose a challenge in itself, see Remark 39). From a categorical perspective, we might describe this issue as the fact that this construction has no universal property: how long you run the construction before stopping is not uniquely determined by the input morphism f. This is what was noticed and rectified in [13].

4 Natural weak factorization systems

Natural weak factorization systems (NWFSs) are an algebraic refinement of WFSs due to Grandis and Tholen [14]. An NWFS is based on a *functorial* factorization, which gives a canonical, well-behaved choice for the factorizations, as opposed to the structureless factorization in a WFS, see Definition 15. We impose additional algebraic structure, making it so that being an $\mathcal{L}$- or $\mathcal{R}$-map is no longer a *property* like for WFSs, but an algebraic *structure* on morphisms. This fixes the constructive issues in the closure properties of WFSs.

4.1 Functorial factorizations

Recall Example 6, defining $\mathcal{C}^\mathbf{3}$ as a displayed category over $\mathcal{C}^\mathbf{2}$.

▶ **Definition 25** (`functorial_factorization`)**.** *A functorial factorization F over a category $\mathcal{C}$ is a section from $\mathcal{C}^\mathbf{2}$ to $\mathcal{C}^\mathbf{3}$.*

▶ Remark 26. Compare this with the definition of factorization Definition 15. That was a section of the composition *function* $\operatorname{ob}\mathcal{C}^\mathbf{3} \to \operatorname{ob}\mathcal{C}^\mathbf{2}$, as opposed to the projection *functor* $\mathcal{C}^\mathbf{3} \longrightarrow \mathcal{C}^\mathbf{2}$.

There are three natural functors $d_0, d_1, d_2 : \mathcal{C}^\mathbf{3} \longrightarrow \mathcal{C}^\mathbf{2}$ which take a composable pair $X \xrightarrow{\lambda_f} E_f \xrightarrow{\rho_f} Y$ to ρ_f, $\lambda_f \cdot \rho_f$, and λ_f, respecively (here d_1 coincides with the canonical projection). We obtain two endofunctors $\mathcal{C}^\mathbf{2} \longrightarrow \mathcal{C}^\mathbf{2}$ by considering $R := F \cdot d_0$ (which sends an f to its right map ρ_f) and $L := F \cdot d_2$ (which sends an f to its left map λ_f).

▶ Remark 27 (The need for sections). With our definition, the left and right functors L and R are automatically compatible. That is to say, for any morphism $f : X \to Y$, *definitional equalities* arising from the definition of a section make for a well-typed (and trivial) equality $L(f) \cdot R(f) =_{X \to Y} f$.

A more naive approach would be to specify F as a section of d_1 in the usual sense:

$$\sum_{F:\mathcal{C}^\mathbf{2} \longrightarrow \mathcal{C}^\mathbf{3}} F \cdot d_1 = \mathrm{id}_{\mathcal{C}^\mathbf{2}} .$$

However, with this definition the composite $L(f) \cdot R(f)$ may not have the same domain and codomain as f. We merely know that their domains and codomains are *propositionally* equal, so the equality $L(f) \cdot R(f) = f$ is now *ill-typed*. One could use the `idtoiso` function [27], mapping equalities of objects to isomorphisms between them, but this does not capture the classical theory, which asks for $L(f) \cdot R(f)$ to be strictly equal, not just isomorphic, to f.

4.2 Natural weak factorization systems

Using only the data of a functorial factorization F, we can view the left and right functors L and R as a copointed endofunctor (L, Φ) and a pointed endofunctor (R, Λ) by defining:

$$\Phi_f := \begin{tikzcd} X \arrow[r, equal] \arrow[d, "\lambda_f"'] & X \arrow[d, "f"] \\ E_f \arrow[r, "\rho_f"'] & Y \end{tikzcd} \quad \text{and} \quad \Lambda_f := \begin{tikzcd} X \arrow[r, "\lambda_f"] \arrow[d, "f"'] & E_f \arrow[d, "\rho_f"] \\ Y \arrow[r, equal] & Y \end{tikzcd} . \tag{2}$$

▶ **Definition 28** (`nwfs`). *A* natural weak factorization system (NWFS) *is given by a functorial factorizaton F, together with an extension of (R, Λ) to a monad $\mathsf{R} = (R, \Lambda, \Pi)$ and the extension of the (L, Φ) to a comonad $\mathsf{L} = (L, \Phi, \Sigma)$. Such an NWFS is said to* lie over *F.*

It will be useful to split this definition into two halves: LNWFS and RNWFS. The former contains only the data concerning the left comonad, the latter contains only the data concerning the right monad. We define the notion of LNWFS, an RNWFS is defined dually.

▶ **Definition 29** (`lnwfs_over`, cf. [13, 4.5]). *The* left half of an NWFS *(LNWFS) is given by a functorial factorization F, with an extension of (L, Φ) to a comonad $\mathsf{L} = (L, \Phi, \Sigma)$.*

4.3 Algebraic structure

Now we form a category $\mathbf{Ff}_{\mathcal{C}}$ of functorial factorizations on $\mathcal{C}$ (`Ff`) by defining morphisms.

▶ **Definition 30** (`fact_mor`, cf. [13, 3.3]). *A* morphism of functorial factorizations *$\tau : F \to F'$ is a natural transformation between sections.*

Since we defined NWFSs as functorial factorizations "with added structure", we define the category of NWFSs on $\mathcal{C}$ as a displayed category over $\mathbf{Ff}_{\mathcal{C}}$. We again split this construction into LNWFSs and RNWFSs, yielding two displayed categories: $\mathbf{LNWFS}_{\mathcal{C}}$ and $\mathbf{RNWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$. Together they form a displayed category $\mathbf{NWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$.

In order to do this, we require some additional structure on the morphisms in $\mathbf{Ff}_{\mathcal{C}}$. Take $F, F' : \mathbf{Ff}_{\mathcal{C}}$. A morphism $\tau : F \to F'$ induces canonical natural transformations $\tau_L : L \Longrightarrow L'$ and $\tau_R : R \Longrightarrow R'$ by *whiskering* with d_2 and d_0.

▶ **Definition 31** (`LNWFS`, `RNWFS`, `NWFS`, cf. [13, 3.3,4.5]). *A morphism $\tau : F \to F'$ in $\mathbf{Ff}_{\mathcal{C}}$ is*

- *a* morphism of LNWFSs *if F and F' underlie LNWFSs and τ_L is a comonad morphism;*
- *a* morphism of RNWFSs *if F and F' underlie RNWFSs and τ_R is a monad morphism;*
- *a* morphism of NWFSs *if F and F' underlie NWFSs and τ is both a morphism of LNWFSs and of RNWFSs.*

These three properties define the displayed morphisms of displayed categories $\mathbf{LNWFS}_{\mathcal{C}}$, $\mathbf{RNWFS}_{\mathcal{C}}$, and $\mathbf{NWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$.

Similar to a WFS, an NWFS (L, R) has left- and right maps. These are defined to be the coalgebras of the comonad L and the algebras of the monad R respectively. We denote the categories of left and right maps of an (L, R) as L-$\mathbf{Map}$ and R-$\mathbf{Map}$ respectively.

4.4 Fixing the constructive issues

The algebraic structure in NWFSs allows us to construct fillers in lifting problems between any L-$\mathbf{Map}$ and R-$\mathbf{Map}$, fixing the constructive issues in the theory of WFSs.

▶ **Lemma 32** (`L_map_R_map_elp`, cf. [13, 2.15]). *Let (L, R) be an NWFS over F, $f : X \to Y$ an L-$\mathbf{Map}$, $g : A \to B$ an R-$\mathbf{Map}$. There exists a filler for any (f, g)-lifting problem (h, k).*

Proof. The (co)algebra axioms force the (co)algebra $\alpha_f : f \to \lambda_f$ and $\alpha_g : \rho_g \to g$ to be of specific forms. Specifically, they ensure that the morphisms $X \to X$ and $B \to B$ are in fact identities in the left-hand diagrams below. Consider then the right-hand diagram, obtained by applying F to (h, k) and attaching the (co)algebra morphisms. The filler $Y \to A$ can be read off the diagram as $s \cdot F(h, k)_{11} \cdot p$.

$$\begin{tikzcd}
X \arrow[r, equal] \arrow[d, "f"'] \arrow[dr, phantom, "\alpha_f"] & X \arrow[d, "\lambda_f"] \\
Y \arrow[r, "s"'] & E_f \\
E_g \arrow[r, "p"] \arrow[d, "\rho_g"'] \arrow[dr, phantom, "\alpha_g"] & A \arrow[d, "g"] \\
B \arrow[r, equal] & B
\end{tikzcd}
\rightsquigarrow
\begin{tikzcd}
X \arrow[r, equal] \arrow[d, "f"'] \arrow[dr, phantom, "\alpha_f"] & X \arrow[r, "h"] \arrow[d, "\lambda_f"'] & A \arrow[d, "\lambda_g"] & \\
Y \arrow[r, "s"'] & E_f \arrow[r, "F(h,k)_{11}"] \arrow[d, "\rho_f"'] & E_g \arrow[r, "p"] \arrow[d, "\rho_g"] \arrow[dr, phantom, "\alpha_g"] & A \arrow[d, "g"] \\
& Y \arrow[r, "k"'] & B \arrow[r, equal] & B
\end{tikzcd}$$
◀

▶ Remark 33. Note that we construct the actual filler, and not just the *mere existence* of one. This is an important difference with WFSs, where we only know of the *mere existence* of a filler. We get a canonical choice function for the filler in any given lifting problem between an L-**Map** and an R-**Map**, fixing the problems we had with plain WFSs, see Remark 19. It allows us to prove analogues of the desired closure properties of WFSs, like the following.

▶ **Lemma 34** (`nwfs_closed_coproducts`). *Let I be a set and $\{ f_i : X_i \to Y_i \}_{i:I}$ a family of maps, such that f_i is an* L-**Map** *for every $i : I$. Then $\bigsqcup_{i:I} f_i$ is also an* L-**Map**.

5 The algebraic small object argument

The construction of the ASOA is inductive like its classical counterpart, the SOA. At each step, we construct an object of $\mathbf{LNWFS}_{\mathcal{C}}$. We then apply a general transfinite construction [20], giving us a full NWFS. There will be many similarities between the constructions, but also some obvious differences. The construction also resolves the constructive and categorical issues that we touched upon in Remark 22 and Remark 24.

We will be following Garner [13, 12], rephrasing the theory using univalent foundations and redefining part of the construction to be more direct and fit for formalization.

5.1 The one-step comonad

Let $f : X \to Y$ be a morphism in $\mathcal{C}$. Recall the first step from the iterative process in the small object argument in equation (1). This construction is in fact functorial, yielding a functorial factorization F^1: the *one-step factorization* (`one_step_factorization`).

There is always a natural transformation $\Sigma^1 : L^1 \Longrightarrow L^1 \cdot L^1$, extending (L^1, Φ^1) to a comonad (where $L^1 := F^1 \cdot d_2$, the left part of F^1, and Φ^1 is the copoint of L^1 given in 2), giving us an object of $\mathbf{LNWFS}_{\mathcal{C}}$: the *one-step comonad* (`one_step_comonad`), c.f. [12, Section 5.2]. The pointed endofunctor (R^1, Λ^1) does *not*, in general, extend to a monad, so we do not yet obtain an object of $\mathbf{NWFS}_{\mathcal{C}}$.

The first step in the algebraic small object argument corresponds with the first step in the classical counterpart. The "left part" of the initial factorization already satisfies the properties we need, while the "right part" of the first step has to be "fixed".

5.2 Monoidal categories

Now we construct an NWFS from our one-step comonad L^1. This uses Kelly's *free monoid construction*. In [13], this takes place in a *strict monoidal category*. We instead use a more general notion of *weak* monoidal categories, formalized in `UniMath` in [28]. This is because in `UniMath` it is not possible to sensibly define strict monoidal categories, where associators and unitors are equalities on objects, unless one is working with setcategories (categories whose types of objects are sets). By using weak monoidal categories, our construction applies to more general categories (in particular, univalent categories).

Garner in fact uses *two-fold monoidal categories*, which comprise two interacting monoidal structures. This fits the theory perfectly, but is not yet formalized in `UniMath`, so we formalize only one monoidal structure. We only need one result relating to the other, which we prove directly (`LNWFS_comon_structure_whiskercommutes`). This simplifies our construction.

5.2.1 The monoidal structure on $\mathbf{LNWFS}_{\mathcal{C}}$

The idea behind the construction is to define a monoidal structure on $\mathbf{Ff}_{\mathcal{C}}$, such that a monoid corresponds with an object in $\mathbf{RNWFS}_{\mathcal{C}}$. This monoidal structure lifts to one on $\mathbf{LNWFS}_{\mathcal{C}}$, so that a monoid in $\mathbf{LNWFS}_{\mathcal{C}}$ corresponds with an object of $\mathbf{NWFS}_{\mathcal{C}}$. We define the unit of the monoidal structure on $\mathbf{Ff}_{\mathcal{C}}$ to be the initial object I mapping

$$X \xrightarrow{f} Y \quad \mapsto \quad X \xrightarrow{\mathrm{id}_X} X \xrightarrow{f} Y. \qquad (\texttt{Ff\_lcomp\_unit}, \text{ cf. [13, Theorem 4.14]})$$

For two functorial factorizations F, F', we define their tensor product $F' \otimes F$ to be

$$X \xrightarrow{f} Y \quad \mapsto \quad X \xrightarrow{\lambda_f \cdot \lambda'_{\rho_f}} E'_{\rho_f} \xrightarrow{\rho'_{\rho_f}} Y. \qquad (\texttt{Ff\_lcomp}, \text{ cf. [13, Theorem 4.14]})$$

▶ **Lemma 35** (`Ff_monoidal`, `Ff_monoid_is_RNWFS`, cf. [13, Theorem 4.14]). *The pair* $(\otimes, I)$ *defines a monoidal structure on* $\mathbf{Ff}_{\mathcal{C}}$*. A monoid structure on* $F : \mathbf{Ff}_{\mathcal{C}}$ *corresponds to an object of* $\mathbf{RNWFS}_{\mathcal{C}}$ *over* F*.*

Noting how $\otimes$ acts on the right functor, the second claim boils down to the fact that a monad is a monoid in the category of endofunctors. Garner mentions the lifting of the monoidal structure to $\mathbf{LNWFS}_{\mathcal{C}}$ in the more general setting of two-fold monoidal categories [13, 4.11], but in the absence of this theory in `UniMath`, we take a direct approach. Proving this took about 1000 lines of formalization (`LNWFSMonoidalStructure.v`), and is the file that takes the longest to compile on various setups (see for example the discussion in `PR 1858`).

▶ Remark 36. The machinery used to lift the monoidal structure on $\mathbf{Ff}_{\mathcal{C}}$ to one on $\mathbf{LNWFS}_{\mathcal{C}}$ is that of *displayed monoidal categories* [3, `disp_monoidal`]. It allows one to define a monoidal structure on (the total category of) a displayed category over some monoidal category, by defining diplayed analogues of the monoidal data in the base category.

▶ **Lemma 37** (`LNWFS_tot_monoidal`, `LNWFS_tot_monoid_is_NWFS`, cf. [13]). *Let* L, L' : $\mathbf{LNWFS}_{\mathcal{C}}$ *over* F, F' : $\mathbf{Ff}_{\mathcal{C}}$ *respectively. Then there is an LNWFS structure on* $F \otimes F'$*. There is also an LNWFS structure on* I*, lifting* $(\otimes, I)$ *to a monoidal structure on* $\mathbf{LNWFS}_{\mathcal{C}}$*. Furthermore, a monoid* L : $\mathbf{LNWFS}_{\mathcal{C}}$ *over some* F : $\mathbf{Ff}_{\mathcal{C}}$ *corresponds with an object of* $\mathbf{NWFS}_{\mathcal{C}}$ *over* F*.*

▶ Remark 38. The classical small object argument boils down to a transfinite tensor product

$$L^\alpha := L^1 \otimes L^1 \otimes \ldots \otimes L^1 : f \mapsto \lambda_f^\alpha.$$

This is not satisfactory, as it leaves us with the same issues discussed before, see Remark 24. We fix this by defining the iterative step with a coequalizer, associating duplicate cells (`next_pair_diagram_coeq`), and a simple convergence condition, removing the need for arbitrary truncation (`T_preserves_diagram_on`).

5.3 The iterative step

Garner generalized a transfinite construction by Kelly [20] to generate a monoid in a monoidal category $\mathcal{V}$, given certain "smallness requirements" on a generating object $T : \mathcal{V}$ and on $\mathcal{V}$ itself. The construction defines a sequence indexed by the category of small ordinals [13, 4.16], converging at some limit ordinal.

▶ Remark 39 (Limitations of ordinals in `UniMath`). What limit ordinal Garner's generalized sequence converges at [13, 4.16] is dictated by the hypothesis in [13, Proposition 4.19], requiring that "$T\otimes(-)$ preserves λ-filtered colimits", which is reduced to smallness requirement (*) in [13]. We limit ourselves to the first (finitely filtered) limit ordinal ω by replacing (*) with finite presentability, and substituting the hypothesis on T with **(V3)**.

We do this since the theory of (filtered) ordinals has not been developed enough in `UniMath` (or HoTT in general [9]). This is still sufficient to apply the theorem to important examples in, for instance, **SSET** and [5].

Formalizing requirement [13, (†)] or other ordinals should only involve adapting the proofs in (`GenericFreeMonoidSequence.v`), most notably up to the convergence of the sequence (`T_preserves_diagram_impl_convergence_on`), with appropriate hypotheses.

5.3.1 The transfinite sequence

For this section, we assume $\mathcal{V}$ to be a monoidal category that has all connected colimits and T to be a pointed object in $\mathcal{V}$, with point $t : I \to T$. In the algebraic small object argument, $\mathcal{V}$ will be $\mathbf{LNWFS}_{\mathcal{C}}$ and T will be L^1. It is easier and more performant to define this sequence on an abstract monoidal category in formalization, but it is useful to keep our main application in mind, particularly in the cases of **TOP** or **SSET**.

We assume the following "smallness requirements" on $\mathcal{V}$ and T.

(V1) $\mathcal{V}$ has ω-colimits and coequalizers.
(V2) $\mathcal{V}$ is *right closed* (so the functor $(-)\otimes A$ preserves colimits for all $A : \mathcal{V}$).
(V3) The functor $T\otimes(-)$ preserves ω-colimits and coequalizers.

Given objects $X_0 := A, X_1 := T\otimes A$ in $\mathcal{V}$ and $\sigma_0 := \mathrm{id}_{T\otimes A} : T\otimes X_0 \to X_1$, we inductively define a transfinite sequence, called the *free T-algebra sequence for A* [13, 4.16]. For a successor ordinal $\alpha+ := \alpha + 1$ we define $X_{\alpha++}$ and $\sigma_{\alpha+} : T\otimes X_{\alpha+} \to X_{\alpha++}$ as the following coequalizer:

$$\begin{tikzcd}
 & X_{\alpha+} \arrow[dr, "t\otimes X_{\alpha+}"] & & \\
T\otimes X_\alpha \arrow[ur, "\sigma_\alpha"] \arrow[dr, "T\otimes(t\otimes X_\alpha)"'] & & T\otimes X_{\alpha+} \arrow[r, "\sigma_{\alpha+}"] & X_{\alpha++} \\
 & T\otimes(T\otimes X_\alpha) \arrow[ur, "T\otimes\sigma_\alpha"'] & &
\end{tikzcd}$$

For any step α, we define $x_\alpha : X_\alpha \to X_{\alpha+}$ to be $(t\otimes X)\cdot\sigma_\alpha$. The full sequence becomes

$$\begin{tikzcd}
 & T\otimes X_0 \arrow[d, "\sigma_0"] & T\otimes X_1 \arrow[d, "\sigma_1"] & \dots & & T\otimes X_\alpha \arrow[d, "\sigma_\alpha"] & \\
X_0 \arrow[ur, "t\otimes X_0"] \arrow[r, "x_0"'] & X_1 \arrow[ur] \arrow[r, "x_1"'] & X_2 \arrow[ur] \arrow[r, "x_2"'] & \dots \arrow[r] & X_\alpha \arrow[ur, "t\otimes X_\alpha"] \arrow[r, "x_\alpha"'] & X_{\alpha+} \arrow[r] & \dots
\end{tikzcd} \tag{3}$$

▶ Remark 40. The morphism $t\otimes X_\alpha$ is actually a morphism $I\otimes X_\alpha \to T\otimes X_\alpha$, so the diagram is actually ill-typed. By definition, there is a natural isomorphism to correct for this. Morphisms like this one are left out for simplicity, reading closer to the notion of *strict monoidal categories*, but they are accounted for in the formalization.

▶ Remark 41. In the examples in **TOP** and **SSET**, the functor $T\otimes(-)$ corresponds to "gluing cells". Then X_α corresponds to "α steps of gluing cells to A, *without duplicates*." The coequalizers σ_α are continuous maps that identify duplicate cells with ones glued previously.

The sequence is defined inductively, using the previous *two* objects and the previous morphism to define the next morphism and object. In order to do this properly, with *definitional* equalities, we introduce a helper type, capturing the data of one "triangle" in the sequence.

▶ **Definition 42** (`pair_diagram`). *A "pair diagram", corresponding to the "triangle" of step α in the sequence displayed in equation* (3)*, is an object of $\mathcal{C}^{\mathbf{3}}$ of the form*

$$X_\alpha \xrightarrow{t \otimes X_\alpha} T \otimes X_\alpha \xrightarrow{\sigma_\alpha} X_{\alpha+}.$$

▶ Remark 43. The only real data in this object are X_α, $X_{\alpha+}$ and $\sigma_\alpha : T \otimes X_\alpha \to X_{\alpha+}$.

Indeed, one can define the $(\alpha+1)$-th pair diagram using only the data from the α-th pair diagram. The inductive definition allows us to make sure left object of the $(\alpha+1)$-th pair diagram is in fact *definitionally equal* to the right object of the α-th pair diagram. Assumptions **(V1)** and **(V3)** ensure this sequence converges (`T_preserves_diagram_impl_convergence_on`), cf. [13, Proposition 4.17]. The limit when $A := I$ yields a *T-algebra* (T^∞, τ^∞), consisting of an object $T^\infty : \mathcal{V}$ and a morphism $\tau^\infty : T \otimes T^\infty \to T^\infty$, which we will show is a monoid.

▶ Remark 44. Intuitively, keeping **TOP** or **SSET** in mind, we may view the object T^∞ as the "space with *all* cells attached". The T-algebra map τ^∞ describes how one more step of attaching cells (through tensoring with T) can be collapsed back into T^∞ itself.

5.4 Obtaining the free monoid

In [12, Proposition 27], the forgetful functor from the category of T-algebras to $\mathcal{V}$ is used to obtain a monoid. Instead, we define the monoid structure more directly, allowing for a much more direct and intuitive construction. There is an obvious choice for the unit $\eta^\infty : I \to T^\infty$: the canonical inclusion into the colimit $X_0 \hookrightarrow T^\infty$. It remains to find a multiplication. By assumption **(V2)**, we have

$$T^\infty \otimes T^\infty \cong \operatorname{colim}(X_\alpha \otimes T^\infty).$$

We define the multiplication $\mu^\infty : T^\infty \otimes T^\infty \to T^\infty$ by defining a family of morphisms $\{ \tau_\alpha : X_\alpha \otimes T^\infty \to T^\infty \}$ that forms a cocone on $\{ X_\alpha \otimes T^\infty \}$.

▶ **Lemma 45** (`Tinf_pd_Tinf_map`). *There is a family of maps $\{ \tau_\alpha : X_\alpha \otimes T^\infty \to T^\infty \}$ such that the following diagram commutes for any α.*

$$\begin{tikzcd}
T \otimes X_\alpha \otimes T^\infty \arrow[r, "\sigma_\alpha \otimes T^\infty"] \arrow[d, "T \otimes \tau_\alpha"'] & X_{\alpha+} \otimes T^\infty \arrow[d, "\tau_{\alpha+}"] \\
T \otimes T^\infty \arrow[r, "\tau^\infty"'] & T^\infty
\end{tikzcd}$$

▶ Remark 46. Intuitively, the τ_α "collapse α steps of gluing cells into T^∞". The diagram tells us that it does not matter if we first collapse α steps of cells into T^∞, and then the last step, or if we first collapse the last step into the first α steps, and then collapse that into T^∞.

We define the τ_α inductively (`free_monoid_coeq_sequence_on_Tinf_pd_Tinf_map`), with obvious choices for τ_0 and τ_1. In the inductive step, we use assumption **(V2)** to define $\tau_{\alpha++}$ as the unique map out of the coequalizer

$$\begin{tikzcd}
 & X_{\alpha+} \otimes T^\infty \arrow[dr, "t \otimes X_{\alpha+} \otimes T^\infty"] & & & X_{\alpha++} \otimes T^\infty \arrow[dd, dashed, "\exists! \tau_{\alpha++}"] \\
T \otimes X_\alpha \otimes T^\infty \arrow[ur, "\sigma_\alpha \otimes T^\infty"] \arrow[dr, "T \otimes t \otimes X_\alpha \otimes T^\infty"'] & & T \otimes X_{\alpha+} \otimes T^\infty \arrow[urr, "\sigma_{\alpha+} \otimes T^\infty"] \arrow[dr, dashed, "T \otimes \tau_{\alpha+}"] & & \\
 & T \otimes T \otimes X_\alpha \otimes T^\infty \arrow[ur, "T \otimes \sigma_\alpha \otimes T^\infty"'] & & T \otimes T^\infty \arrow[r, dashed, "\tau^\infty"] & T^\infty
\end{tikzcd}$$

The required commutativity constraint on $\tau_{\alpha++}$ can be read off the diagram directly. The smallness assumptions allow one to show that $(T^\infty, \eta^\infty, \mu^\infty)$ is indeed a monoid in $\mathcal{V}$ (`Tinf_monoid`).

5.5 Smallness requirements on $\mathbf{LNWFS}_{\mathcal{C}}$ and L^1

We used assumptions on $\mathcal{V}$ and T that we now need to show for $\mathcal{V} = \mathbf{LNWFS}_{\mathcal{C}}$ and $T = L^1$. We show **(V1)** and **(V2)** and reduce **(V3)** to a much weaker requirement, only involving the subclass of morphisms J used to define L^1. In this section, we do not have access in `UniMath` to some of the categorical machinery (e.g., the full theory of (co)ends) used in [13], so our arguments are more direct.

5.5.1 Cocompleteness of $\mathbf{LNWFS}_{\mathcal{C}}$

Issues commonly arise in the displayed nature of $\mathbf{LNWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$, or that of $\mathcal{C}^{\mathbf{3}}$ over $\mathcal{C}^{\mathbf{2}}$, requiring definitional equalities where naive arguments only produce propositional ones. Consider the image of a morphism $f : X \to Y$ under an ω-chain of functorial factorizations $\{ F_\alpha \}$, as well as its colimit, in the left-hand diagram below.

$$\begin{tikzcd}
X \arrow[r, equal] \arrow[d] & X \arrow[r, equal] \arrow[d] & X \arrow[r, dotted] \arrow[d] & X^\infty \arrow[d] & & & X \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr, dotted] & & \\
E_f^0 \arrow[r] \arrow[d] & E_f^1 \arrow[r] \arrow[d] & E_f^2 \arrow[r, dotted] \arrow[d] & E_f^\infty \arrow[d] & \rightsquigarrow & E_f^0 \arrow[r] \arrow[dr] & E_f^1 \arrow[r] \arrow[d] & E_f^2 \arrow[r, dotted] \arrow[dl] & \cdots \arrow[dll, dotted] \\
Y \arrow[r, equal] & Y \arrow[r, equal] & Y \arrow[r, dotted] & Y^\infty & & & Y & &
\end{tikzcd}$$

The colimit exists (`three_colims`), but its domain X^∞ and codomain Y^∞ need not be *definitionally equal* to X and Y respectively. We merely know they are isomorphic. We could correct the domain and codomain with these isomorphisms to define an object of $\mathcal{C}^{\mathbf{3}}$ over f, and in turn a colimit $F_\infty : \mathbf{Ff}_{\mathcal{C}}$. However, this is quite cumbersome to work with as we want to define a comonad structure on the left functor of F_∞ to define colimits in $\mathbf{LNWFS}_{\mathcal{C}}$.

Instead, recall what the actual data is in functorial factorizations and transformations between them: the middle objects in the image, and the morphisms between them. We "collapse" the (definitional) equalities in the left-hand diagram to form the right-hand diagram above. Let E_f^∞ be the colimit of the E_f^α. We always get a map $E_f^\infty \to Y$, but a map $X \to E_f^\infty$ can only be defined when the colimit is non-empty and connected, namely as the canonical inclusion of $X \to E_f^\alpha \to E_f^\infty$ for an arbitrary α. Indeed, we have the following.

▶ **Lemma 47** (`ColimFfCocone`, `ColimLNWFSCocone`, cf. [13, Prop. 4.18])**.** *Both* $\mathbf{Ff}_{\mathcal{C}}$ *and* $\mathbf{LNWFS}_{\mathcal{C}}$ *have all connected, non-empty colimits, where colimits in* $\mathbf{Ff}_{\mathcal{C}}$ *are constructed as described above, and colimits in* $\mathbf{LNWFS}_{\mathcal{C}}$ *lie over those of the projected diagrams in* $\mathbf{Ff}_{\mathcal{C}}$*.*

5.5.2 Right closure of $\mathbf{LNWFS}_{\mathcal{C}}$

Here too, Garner uses a high level argument [13, Proposition 4.18], but we take a more direct approach. One can show that the functor $(-) \otimes A : \mathbf{Ff}_{\mathcal{C}} \longrightarrow \mathbf{Ff}_{\mathcal{C}}$ preserves colimits for any $A : \mathbf{Ff}_{\mathcal{C}}$ quite easily. The following lemma then proves requirement **(V2)**.

▶ **Lemma 48** (`Ff_iso_inv_LNWFS_mor`)**.** *Let* $L, L' : \mathbf{LNWFS}_{\mathcal{C}}$ *over* $F, F' : \mathbf{Ff}_{\mathcal{C}}$ *respectively. Let* $\tau : F \to F'$ *be an isomorphism. Then* τ^{-1} *is a morphism of LNWFSs whenever* τ *is.*

5.5.3 Reducing the smallness requirement on L^1

To reduce requirement **(V3)** to a simpler one, we mostly follow [12, Proposition 32]. The last part of this reduction has again been rephrased to be more direct and fit for formalization, using low level arguments (`OneStepMonadSmall.v`). In the end, the smallness requirement we are left with is phrased in terms of *presentable* objects in a category.

▶ **Definition 49** (`presentable`). *Let $X : \mathcal{C}$. Then X is called* ω-presentable *if and only if the covariant hom-set functor* $\hom(X, -) : \mathcal{C} \longrightarrow \mathbf{SET}$ *preserves ω-colimits.*

▶ **Theorem 50** (`small_object_argument`). *Let J be a subclass of morphisms in a cocomplete category $\mathcal{C}$, such that any $g \in J$ is ω-presentable in $\mathcal{C}^{\mathbf{2}}$. Then there exists an NWFS in $\mathcal{C}$.*

▶ Remark 51. Besides using arbitrary ordinals and not just ω, Garner describes another, more involved smallness requirement [13, (†), Proposition 4.22]. Still, the case where we work is sufficient for us. See also Remark 39.

6 Scaling

Besides the strategies we used to make our formalization mathematically feasible, we also used the following strategies to make our formalization computationally feasible.

- Proper use of abstraction: Ending a lemma with `Qed`, makes it *opaque*, preventing the proof checker from *unfolding* it in other proofs. This significantly speeds up the proof checker, and the formalization process as a whole. The `abstract` tactic allows one to construct opaque terms within a proof. This alone sped up the compile time for the (`FFMonoidalStructure.v`) file from 30 minutes to 30 seconds on one setup.
- Avoiding `rewrite`: Though useful, this tactic produces large and unwieldy proof terms, which take a long time to verify. Instead we often used the `etrans` and `apply` tactics.
- Removing `cbn`, `simpl`, `unfold` or other "unfolding" tactics from finished proofs: These unfolded terms take much longer to type check. Avoiding the `rewrite` tactic allows us to completely remove these tactics from finished proofs, as tactics like `etrans` and `apply` do not consider the precise syntactic form of a goal term, but only consider its value up to definitional equality.
- Sectioning and local opacity: Compile times were also reduced by using context variables and proper sectioning. Local opacity (through the `Opaque` vernacular, used in e.g. (`LNWFSClosed.v`)) provided the benefits of opaque proof terms when the precise definition of a certain construction was not needed in a file, without enforcing opacity globally.

7 Conclusion

We have rephrased and formalized Garner's algebraic small object argument [13] using machinery more appropriate for formalization in `UniMath`, like displayed categories and weak monoidal categories.

Let us briefly go over some of the main differences in the argument by Garner and this work. First, we filled in many details which [13] left implicit. For example, the explicit construction of lifting of the monoidal structure on $\mathbf{Ff}_{\mathcal{C}}$ to $\mathbf{LNWFS}_{\mathcal{C}}$ was left out in [13], but took over 1000 lines of formalization and is the file that takes the longest to compile in the entire formalization.

Secondly, we introduced more modern language, in the form of displayed categories [2] and a weak notion of monoidal categories [28].

Thirdly, we left out a lot of complex theory that Garner uses. This is, again, partly due to the limited available results in `UniMath`, but it contributes to the accessibility of the proofs. Complex constructions like two-fold monoidal categories are left out, Garner's construction of the free monoid is replaced with a more direct and intuitive one.

The formalization gave more insight into the details of the theory, pointing out constructive issues in the theory of WFSs and showing how few assumptions Garner's algebraic small object argument really needs. In the formalization of the construction, we never assumed any categories to be setcategories (as they are in any classical theory, including [13]) or univalent categories.

There is still some work that may be done in the formalization of Garner's article, for example overcoming our limitations mentioned in Remark 39 and Remark 51. Other than that, there are further results beyond the main theorem of [13] that could be formalized, for instance [13, Proposition 5.4]. Some theory on this has already been formalized (`algebraically_free`), but once complete more examples could be worked out as well. Beyond that, our ultimate goal is to use this to formalize semantics of HoTT/UF within `UniMath`.

References

1 Benedikt Ahrens, Kapulkin Krzysztof, and Michael Shulman. Univalent categories and the rezk completion. *Mathematical Structures in Computer Science*, 25(5):1010–1039, 2015. doi:10.1017/S0960129514000486.

2 Benedikt Ahrens and Peter LeFanu Lumsdaine. Displayed categories. *Logical Methods in Computer Science*, Volume 15, Issue 1, May 2017. doi:10.23638/LMCS-15(1:20)2019.

3 Benedikt Ahrens, Ralph Matthes, and Kobe Wullaert. Formalizing monoidal categories and actions for syntax with binders, 2023. arXiv:2307.16270.

4 Benedikt Ahrens, Paige Randall North, Michael Shulman, and Dimitris Tsementzis. The Univalence Principle, 2022. arXiv:2102.06275.

5 Steve Awodey. A cubical model of homotopy type theory, 2016. arXiv:1607.06413.

6 Steve Awodey. Cartesian cubical model categories, 2023. arXiv:2305.00893.

7 Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Math. Proc. Cambridge Philos. Soc.*, 146(1):45–55, 2009. doi:10.1017/S0305004108001783.

8 Evan Cavallo, Anders Mörtberg, and Andrew W Swan. Unifying Cubical Models of Univalent Type Theory. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2020.14.

9 Tom de Jong, Nicolai Kraus, Fredrik Nordvall Forsberg, and Chuangjie Xu. Set-theoretic and type-theoretic ordinals coincide. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, June 2023. doi:10.1109/lics56636.2023.10175762.

10 Nicola Gambino and Simon Henry. Towards a constructive simplicial model of Univalent Foundations. *Journal of the London Mathematical Society*, 105(2):1073–1109, 2022. doi:10.1112/jlms.12532.

11 Nicola Gambino and Marco Federico Larrea. Models of Martin-Löf type theory from algebraic weak factorisation systems. *The Journal of Symbolic Logic*, 88(1):242–289, June 2021. doi:10.1017/jsl.2021.39.

12 Richard Garner. Cofibrantly generated natural weak factorisation systems, 2007. arXiv:math/0702290.

13 Richard Garner. Understanding the small object argument. *Applied Categorical Structures*, 17(3):247–285, April 2008. doi:10.1007/s10485-008-9137-4.

14 Marco Grandis and Walter Tholen. Natural weak factorization systems. *Archivum Mathematicum*, 42, January 2006.

15 Dennis Hilhorst. A Formalization of the Algebraic Small Object Argument in UniMath, November 2023. Available at https://studenttheses.uu.nl/handle/20.500.12932/45658.

16 Mark Hovey. *Model Categories*. Mathematical surveys and monographs. American Mathematical Society, 2007.

17 André Joyal. Notes on quasi-categories. *Preprint*, 2008.

18 André Joyal and Myles Tierney. Strong stacks and classifying spaces. In Aurelio Carboni, Maria Cristina Pedicchio, and Guiseppe Rosolini, editors, *Category Theory*, pages 213–236, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

19 Chris Kapulkin and Peter LeFanu Lumsdaine. The Simplicial Model of Univalent Foundations (after Voevodsky), 2018. `arXiv:1211.2851`.

20 G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bulletin of the Australian Mathematical Society*, 22(1):1–83, 1980. `doi:10.1017/S0004972700006353`.

21 Jacob Lurie. Derived Algebraic Geometry III: Commutative Algebra, 2009. `arXiv:math/0703204`.

22 J. P. May and K. Ponto. *More Concise Algebraic Topology: Localization, Completion, and Model Categories*. University of Chicago Press, Chicago, 2011. URL: `https://cds.cern.ch/record/1416976`.

23 Fabien Morel and Vladimir Voevodsky. A^1-homotopy theory of schemes. *Publications Mathématiques de l'IHÉS*, 90:45–143, 1999.

24 nLab authors. weak factorization system. Available at `https://ncatlab.org/nlab/show/weak+factorization+system`, March 2024. Revision 46.

25 Daniel G. Quillen. *Homotopical Algebra*. Lecture notes in mathematics. Springer-Verlag, 1967.

26 Charles Rezk. A model for the homotopy theory of homotopy theory. *Transactions of the American Mathematical Society*, 353(3):973–1007, 2001.

27 Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. Unimath — a computer-checked library of univalent mathematics. Available at `http://unimath.org`. `doi:10.5281/zenodo.7848572`.

28 Kobe Wullaert, Ralph Matthes, and Benedikt Ahrens. Univalent Monoidal Categories. In Delia Kesner and Pierre-Marie Pédrot, editors, *28th International Conference on Types for Proofs and Programs (TYPES 2022)*, volume 269 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:21, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.TYPES.2022.15`.