A layout algorithm for
higher-dimensional string diagrams

Calin Tataru

Computer Laboratory
University of Cambridge

calin.tataru@cl.cam.ac.uk

Jamie Vicary

Computer Laboratory
University of Cambridge

jamie.vicary@cl.cam.ac.uk

The algebraic zigzag construction has recently been introduced as a combinatorial foundation for a higher dimensional notion of string diagram. For use in a proof assistant, a layout algorithm is required to determine the optimal rendering coordinates, across multiple projection schemes including 2D, 3D, and 4D. For construction of these layouts, a key requirement is to determine the linear constraints which the geometrical elements must satisfy in each dimension. Here we introduce a new categorical tool called injectification, which lifts a functorial factorization system on a category to diagrams over that category, and we show that implementing this recursively in the category of finite posets allows us to systematically generate the necessary constraints. These ideas have been implemented as the layout engine of the proof assistant homotopy.io, enabling attractive and practical visualisations of complex higher categorical objects.

1 Introduction

Associative n-categories have been recently proposed [4] as a new semistrict model of n-categories with strictly associative and unital composition in all dimensions. They retain sufficient weak structure, in the form of homotopies of composites, to reasonably conjecture that they are equivalent to weak n-categories. This makes them an attractive language for a proof assistant, because they allow composites to be unique without sacrificing expressiveness. Indeed, the theory is the basis for the proof assistant homotopy.io [3] which allows users to construct and manipulate terms in finitely-presented n-categories.

Terms in the theory, called n-*diagrams*, are inherently geometrical and have a direct interpretation as n-dimensional string diagrams. The proof assistant homotopy.io embraces this, and the only way to view
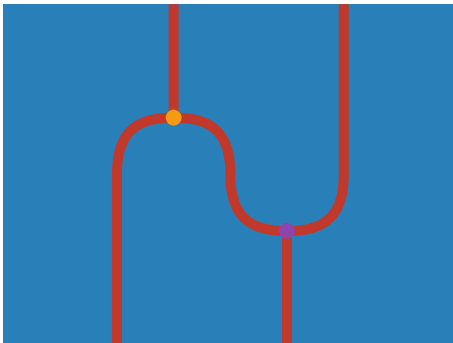
(a) Visual representation
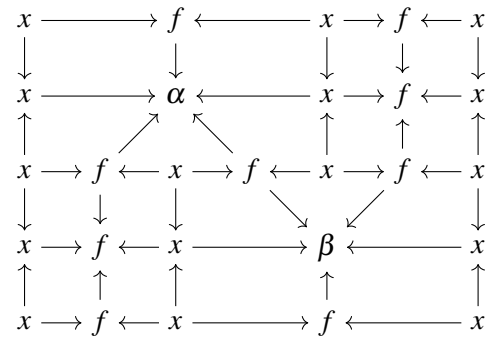
(b) Combinatorial encoding

Figure 1: Example of a 2-diagram in homotopy.io.

or interact with terms is via a visual representation. However, this poses an obstacle, because n-diagrams are combinatorial objects, defined via an inductive construction, which do not contain *layout* information indicating how they should be drawn. Whilst it is easy to obtain a bad layout, getting a good layout that produces nice visuals is a non-trivial problem. Previous proof assistants used ad-hoc methods [1], but these are inefficient and become infeasible in higher dimensions, so a more general solution is needed.

The main contribution of this paper is a new categorical construction called *injectification*, which lifts a functorial factorization system to diagrams valued on a category, in a way which remains functorial at the level of the diagram category. This enables computation of constrains in an abstract way, essentially by taking the colimit of the entire n-diagram itself, valued in posets, where the order relation in the poset captures the constraints that the layout must satisfy. We use this to give a full layout algorithm for string diagrams in arbitrary dimension.

The algorithm has been successfully implemented in the new version of homotopy.io[1], as follows.

- In dimension 2, this enables rendering of planar string diagrams.

- In dimension 3, this allows rendering of 3D diagrams (see Figure 5).

- In dimension 4, this yields rendering of smooth animations of 3D diagrams.

In all these cases, the algorithm produces high-quality output which improves the ability of users to comprehend complex diagrams in higher dimensions. While ad hoc approaches would likely suffice in dimension 2, and with difficulty in dimension 3, the abstract approach we develop here becomes essential from dimension 4 onwards, with the constraint structures becoming highly complex.

1.1   Related work

The theory of associative n-categories was originally developed by Dorn, Douglas, and Vicary [5], and was described in Dorn's thesis [4]. The zigzag construction, which gives an inductive definition of terms in associative n-categories, was described in Reutter and Vicary's paper in LICS 2019 [10] for the purpose of describing a contraction algorithm for building non-trivial terms in the theory. This was followed by Heidemann, Reutter, and Vicary's paper in LICS 2022 [7] on a normalisation algorithm.

There are other proof assistants that use string diagrams as a visualisation method, and all of these must use some layout algorithm (see Table 1). Of these, WIGGLE.PY [2] is perhaps the most similar, since that system also uses linear programming compute the layout; that system is a dedicated 3d renderer and generates the constraints via custom code, whereas we generate the constraints in any number of dimensions via a uniform mathematical approach.

proof assistant	generality	dimensions	layout algorithm
CARTOGRAPHER [12]	symmetric monoidal categories	two	layered graph drawing
QUANTOMATIC [9]	compact closed categories	two	force-based graph drawing
WIGGLE.PY [2]	monoidal bicategories	three	linear programming

Table 1: Comparison of layout algorithms in existing proof assistants.

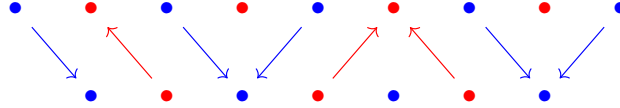[1]Currently in beta stage, the new version of the tool is hosted at `beta.homotopy.io`.

Figure 2: An interleaving of $f : \mathbf{3} \to \mathbf{4}$ in Δ going up, and $\mathsf{R} f : \mathbf{5} \to \mathbf{4}$ in $\Delta_=$ going down.

1.2   Notation

For each $n \in \mathbb{N}$, we write $\mathbf{n}$ for the totally-ordered set $\{0, \ldots, n-1\}$. We write Δ for the category where objects are natural numbers $n \in \mathbb{N}$ and morphisms are monotone maps $\mathbf{n} \to \mathbf{m}$. We also write $\Delta_=$ for the subcategory with non-zero natural numbers and monotone maps preserving the first and last elements.

2   Zigzag construction

We begin by recalling the zigzag construction developed by Reutter and Vicary [10]. This gives a simple inductive definition of *n-diagrams*, which form the terms of the theory of associative *n*-categories.

Definition 2.1. In a category $\mathbf{C}$, a *zigzag X* is a finite sequence of cospans:

$$X(\mathsf{r}0) \longrightarrow X(\mathsf{s}0) \longleftarrow X(\mathsf{r}1) \longrightarrow X(\mathsf{s}1) \longleftarrow \cdots \longrightarrow X(\mathsf{s}n-1) \longleftarrow X(\mathsf{r}n)$$

This consists of *singular objects* of the form $X(\mathsf{s}i)$ for $0 \le i < n$, and *regular objects* of the form $X(\mathsf{r}j)$ for $0 \le j \le n$. Such a zigzag is said to have *length n*, given by the number of singular objects.

There is a natural notion of morphism between zigzags. It is based on an equivalence of categories

$$\mathsf{R} : \Delta \xrightarrow{\sim} \Delta_=^{\mathrm{op}} \tag{1}$$

first observed by Wraith [13], sending n to $n+1$ and each monotone map $f : \mathbf{n} \to \mathbf{m}$ in Δ to the monotone map $\mathsf{R} f : \mathbf{m+1} \to \mathbf{n+1}$ in $\Delta_=$, given by $(\mathsf{R} f)(i) := \min(\{j \in \mathbf{n} \mid f(j) \ge i\} \cup \{n\})$. The equivalence has a nice geometrical intuition, which is illustrated in Figure 2: the map f going up the page is "interleaved" with the map $\mathsf{R} f$ going down the page. We can obtain the definition of a zigzag map from this picture by instantiating the arrows with actual morphisms in $\mathbf{C}$ (for the arrows of f) or $\mathbf{C}^{\mathrm{op}}$ (for the arrows of $\mathsf{R} f$).

Definition 2.2. A *zigzag map $f : X \to Y$* consists of the following, where n is the length of X and m is the length of Y:

- a *singular map* in Δ,

$$f_\mathsf{s} : \mathbf{n} \to \mathbf{m}$$

  with an implied *regular map* in $\Delta_=$,

$$f_\mathsf{r} := \mathsf{R} f_\mathsf{s} : \mathbf{m+1} \to \mathbf{n+1}$$

- for every $0 \le i < n$, a *singular slice* in $\mathbf{C}$,

$$f(\mathsf{s}i) : X(\mathsf{s}i) \to Y(\mathsf{s} f_\mathsf{s}(i))$$

- for every $0 \le j \le m$, a *regular slice* in $\mathbf{C}$,

$$f(\mathsf{r}j) : X(\mathsf{r} f_\mathsf{r}(j)) \to Y(\mathsf{r}j)$$

These must satisfy the following conditions, for every $0 \leq i < n$:

- if $f_{\mathsf{s}}^{-1}(i)$ is empty, this diagram must commute:

$$
Y(\mathsf{r}i) \longrightarrow Y(\mathsf{s}i) \longleftarrow Y(\mathsf{r}i+1)
$$
$$
X(\mathsf{r}\,f_{\mathsf{r}}(i))
$$

- if $f_{\mathsf{s}}^{-1}(i)$ is non-empty, the following diagrams must commute, where p and q are the lower and upper bounds of the preimage, and $p \leq j < q$:

$$
\begin{array}{ccc}
Y(\mathsf{r}i) & \longrightarrow & Y(\mathsf{s}i) \\
\uparrow & & \uparrow \\
X(\mathsf{r}p) & \longrightarrow & X(\mathsf{s}p)
\end{array}
\qquad
\begin{array}{c}
Y(\mathsf{s}i) \\
X(\mathsf{s}j) \longleftarrow X(\mathsf{r}j+1) \longrightarrow X(\mathsf{s}j+1)
\end{array}
\qquad
\begin{array}{ccc}
Y(\mathsf{s}i) & \longleftarrow & Y(\mathsf{r}i+1) \\
\uparrow & & \uparrow \\
X(\mathsf{s}q) & \longleftarrow & X(\mathsf{r}q+1)
\end{array}
$$

Given a category $\mathbf{C}$, we define its *zigzag category* $\mathsf{Zig}(\mathbf{C})$ to be the category whose objects are zigzags and whose morphisms are zigzag maps in $\mathbf{C}$. The construction is functorial in $\mathbf{C}$, so we have a functor

$$
\mathsf{Zig} : \mathbf{Cat} \to \mathbf{Cat} \tag{2}
$$

The construction can be iterated, and we denote the *n-fold zigzag category* by $\mathsf{Zig}^n(\mathbf{C})$.

Note that $\mathsf{Zig}(\mathbf{1})$ is isomorphic to Δ, since a zigzag in the terminal category is uniquely determined by its length. Hence, since every category $\mathbf{C}$ has a unique functor $\mathbf{C} \to \mathbf{1}$, this naturally induces a functor

$$
(-)_{\mathsf{s}} : \mathsf{Zig}(\mathbf{C}) \to \Delta \tag{3}
$$

called the *singular projection*, which sends a zigzag to its length and a zigzag map to its singular map.

2.1   Diagrams from zigzags

Consider a category $\mathbf{C}$ equipped with a *dimension* function $\dim : \mathrm{ob}\,\mathbf{C} \to \mathbb{N}$, such that $\dim(x) < \dim(y)$ whenever there is a non-identity morphism $x \to y$ in $\mathbf{C}$. This is also known as a *direct category*.

Definition 2.3. An *n-diagram* is an object of the *n*-fold zigzag category $\mathsf{Zig}^n(\mathbf{C})$.

We think of the objects of $\mathbf{C}$ as encoding generators in a signature. Then the objects of $\mathsf{Zig}^n(\mathbf{C})$ can be seen as combinatorial encodings of *n*-dimensional string diagrams in the free *n*-category generated by that signature. Not all *n*-diagrams are valid, however, and the full theory of associative *n*-categories has a *typechecking* procedure for checking whether an *n*-diagram is *well-typed* with respect to a signature (see [4, Chapter 8] and [7, Section 7].)

Example 2.4. In the associative 2-category generated by a 0-cell x, a 1-cell $f : x \to x$, and a 2-cell $m : f \circ f \to f$, the 2-cell m can be interpreted as a monad structure, and is encoded by the following 2-diagram:

$$
\begin{array}{ccccc}
x & \longrightarrow & f & \longleftarrow & x \\
\downarrow & & \downarrow & & \downarrow \\
x & \longrightarrow & m & \longleftarrow & x \\
\uparrow & & \nearrow\,\nwarrow & & \uparrow \\
x \longrightarrow f & \longleftarrow & x & \longrightarrow f & \longleftarrow x
\end{array}
$$

Example 2.5. In the associative 3-category generated by a 0-cell x and a pair of 2-cells $\alpha, \beta : \mathrm{id}_x \to \mathrm{id}_x$, the *Eckmann-Hilton move* is a 3-cell that can be interpreted as "braiding x over y", and is encoded by the following 3-diagram:
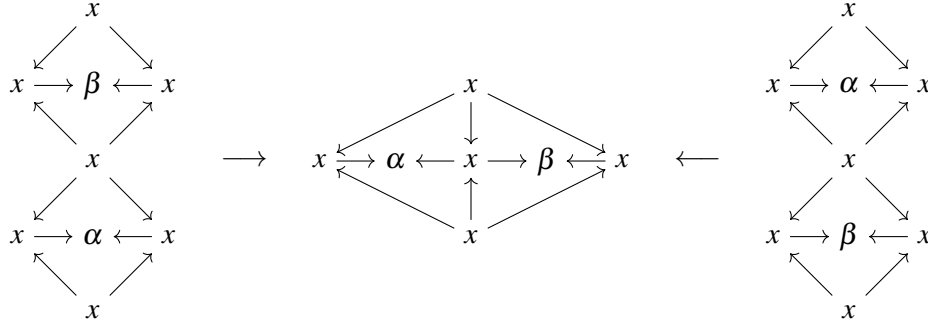
2.2 Constructions

We can define the following constructions on diagrams.

Definition 2.6. Given a zigzag X, its set of *points* is defined as follows, where n is the length of X:

$$\mathsf{pt}(X) = \{\mathsf{s}\,i \mid 0 \leq i < n\} \cup \{\mathsf{r}\,j \mid 0 \leq j \leq n\}$$

Definition 2.7. Given an n-diagram X, its set of *k-points* for $k \leq n$ is defined inductively:

$$\mathsf{pt}^0(X) = \{[\,]\}$$
$$\mathsf{pt}^{k+1}(X) = \{p :: ps \mid p \in \mathsf{pt}(X), ps \in \mathsf{pt}^k(X(p))\}$$

Given a diagram in $\mathsf{Zig}(\mathbf{C})$, it is possible to unfold it into a diagram in $\mathbf{C}$, which is denoted as follows:[2]

$$\frac{X : \mathbf{J} \to \mathsf{Zig}(\mathbf{C})}{X' : \mathsf{pt}(X) \to \mathbf{C}}$$

This construction is called *explosion*. The objects of the category $\mathsf{pt}(X)$ are pairs (i, p) where $i \in \mathbf{J}$ and $p \in \mathsf{pt}(Xi)$, and the arrows are inherited from the zigzags and zigzag maps. By iterating the construction, we can also explode diagrams in the n-fold zigzag category:

$$\frac{X : \mathbf{J} \to \mathsf{Zig}^n(\mathbf{C})}{X^{(k)} : \mathsf{pt}^k(X) \to \mathsf{Zig}^{n-k}(\mathbf{C})}$$

Similarly, the objects of $\mathsf{pt}^k(X)$ are pairs of the form (i, ps) where $i \in \mathbf{J}$ and $ps \in \mathsf{pt}^k(Xj)$.

3 Injectification

In this section, we introduce a categorical construction, called *injectification*, which can be thought of as a generalisation of factorization. We begin by recalling some definitions of factorization systems. [11]

---

[2]This can be made it into a formal adjunction which is described in Appendix A.

3.1 Factorization systems

Definition 3.1. In a category **C**, a morphism f has the *left lifting property* with respect to a morphism g, or equivalently, g has the *right lifting property* with respect to f, if every lifting problem

$$
\begin{array}{ccc}
\bullet & \longrightarrow & \bullet \\
f\downarrow & \nearrow & \downarrow g \\
\bullet & \longrightarrow & \bullet
\end{array}
$$

has a solution, not necessarily unique, making both triangles commute. We denote this by $f \downarrow g$.

Given a class M of morphisms of **C**, we define the following two classes of morphisms:

$$M^{\downarrow} = \{f \mid \forall m \in M.\, m \downarrow f\}$$
$$^{\downarrow}M = \{f \mid \forall m \in M.\, f \downarrow m\}$$

Definition 3.2. A *weak factorization system* (wfs) is a pair $(\mathscr{L}, \mathscr{R})$ of classes of morphisms of **C** so that:

- every morphism f in **C** can be factored as $f = rl$ where $l \in \mathscr{L}$ and $r \in \mathscr{R}$, and
- $\mathscr{L}$ and $\mathscr{R}$ are closed under having the lifting property w.r.t. each other, i.e. $\mathscr{L}^{\downarrow} = \mathscr{R}$ and $^{\downarrow}\mathscr{R} = \mathscr{L}$.

Note that a weak factorization system does not specify *how* to factorize morphisms, merely that such factorizations exist. To give explicit factorization data, we need a functorial factorization structure.

Definition 3.3. A *functorial factorization* on **C** is a section of the composition functor $d_1 : [\mathbf{3}, \mathbf{C}] \to [\mathbf{2}, \mathbf{C}]$.

Explicitly, this specifies how to factorize any morphism:

$$
\begin{array}{ccc}
A & \xrightarrow{\quad f \quad} & B \\
{}_{Lf}\searrow & & \nearrow {}_{Rf} \\
& Ff &
\end{array}
$$

in a way that depends functorially on commutative squares (i.e. the morphisms of $[\mathbf{2}, \mathbf{C}]$):

$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow & & \downarrow \\
C & \xrightarrow{g} & D
\end{array}
\quad \mapsto \quad
\begin{array}{ccccc}
A & \xrightarrow{Lf} & Ff & \xrightarrow{Rf} & B \\
\downarrow & & \vdots & & \downarrow \\
C & \xrightarrow{Lg} & Fg & \xrightarrow{Rg} & D
\end{array}
$$

Definition 3.4. If $(\mathscr{L}, \mathscr{R})$ is a wfs and (F, L, R) is a functorial factorization such that $Lf \in \mathscr{L}$ and $Rf \in \mathscr{R}$ for all f, we say that (F, L, R) is a *functorial realization* of $(\mathscr{L}, \mathscr{R})$. Such a wfs is said to be *functorial*.

Example 3.5. In **Set**, there is a functorial wfs given by:

- $\mathscr{L}$ = monomorphisms and $\mathscr{R}$ = epimorphisms,
- every map $f : A \to B$ is factored through the coproduct:

$$A \rightarrowtail A + B \twoheadrightarrow B$$

Example 3.6. In any topos, there is a functorial wfs given by:

- $\mathscr{L}=$ epimorphisms and $\mathscr{R}=$ monomorphisms,
- every morphism $f : A \to B$ is factored through the *image*

$$A \longrightarrow\!\!\!\!\!\twoheadrightarrow \operatorname{im} f \rightarrowtail\!\!\!\!\longrightarrow B$$

which is the smallest subobject of B for which the factorization exists.

Example 3.7. In **Cat**, there is a functorial wfs given by:

- $\mathscr{L}=$ bijective-on-objects functors and $\mathscr{R}=$ fullly faithful functors,
- every functor $F : A \to B$ is factored through its *full image*

$$A \longrightarrow \overline{\operatorname{im}} F \longrightarrow B$$

which is a category that gets is objects from A and its morphism from B:

$$\operatorname{ob}(\overline{\operatorname{im}} F) = \operatorname{ob} A$$
$$(\overline{\operatorname{im}} F)(a, a') = B(F(a), F(a'))$$

3.2 Main construction

Fix a category **C** and a pair $(\mathscr{L}, \mathscr{R})$ of classes of morphisms of **C**.

Definition 3.8. Given a functor $X : \mathbf{J} \to \mathbf{C}$, an *injectification* of X is a functor

$$\widehat{X} : \mathbf{J} \to \mathbf{C}_{\mathscr{L}}$$

together with a natural transformation whose every component is in $\mathscr{R}$:

$$
\begin{array}{ccc}
\mathbf{J} & \xrightarrow{\quad X \quad} & \mathbf{C} \\
& \widehat{X} \searrow \quad \Uparrow \varepsilon \quad \nearrow & \\
& \mathbf{C}_{\mathscr{L}} &
\end{array}
$$

Conceptually, injectification takes a diagram and produces another diagram of the same shape, where every morphism has been replaced by one in $\mathscr{L}$, together with a way to map back to the original diagram by morphisms in $\mathscr{R}$. The motivating example is when the two classes are (mono, epi), then injectification has the effect of making every morphism injective in a way that preserves the data of the original diagram.

We can also think of injectification as a way to generalise factorizations from morphisms to diagrams. If we let $\mathbf{J} = \mathbf{2}$, then a functor $\mathbf{2} \to \mathbf{C}$ picks out a morphism $f : A \to B$ in **C**, and an injectification can be given by factorizing the morphism f (if such a factorization exists):

$$
\begin{array}{ccc}
A & \xrightarrow{\ f\ } & B \\
\| & & \uparrow{\scriptstyle Rf} \\
A & \xrightarrow[\ Lf\ ]{} & Ff
\end{array}
$$

The main theorem of this section shows that given a wfs, we can use it to construct injectifications of finite poset-shaped diagrams. Moreover, the construction is functorial if the wfs is functorial.

Theorem 3.9. *Assume that:*

- *$(\mathcal{L}, \mathcal{R})$ is a functorial wfs on **C** with functorial realization (F, L, R), and*
- *$\mathbf{C}_{\mathcal{L}}$ has all finite poset-shaped colimits and they are preserved under the inclusion $\mathbf{C}_{\mathcal{L}} \hookrightarrow \mathbf{C}$.*

Then any finite poset-shaped diagram $X : J \to \mathbf{C}$ has an injectification $(\widehat{X}, \varepsilon)$. Moreover, this construction is functorial in that any natural transformation $\alpha : X \Rightarrow Y$ can be lifted to a commutative square in $\mathbf{C}^J$:

$$
\begin{array}{ccc}
X & \xrightarrow{\ \alpha\ } & Y \\
\varepsilon_X \Big\uparrow & & \Big\uparrow \varepsilon_Y \\
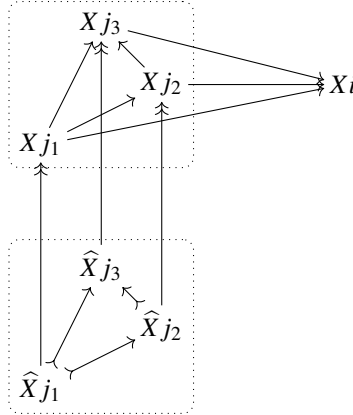\widehat{X} & \dashrightarrow[\widehat{\alpha}] & \widehat{Y}
\end{array}
$$

Proof. We construct the injectification by induction. First, for any minimal $i \in J$, define

$$
\widehat{X}_i := X_i
$$

$$
\varepsilon_i := \mathrm{id}_{X_i}
$$

For any non-minimal $i \in J$, restrict to the downward-closed subset

$$
J \downarrow i = \{ j \in J : j < i \}.
$$

By induction, the injectification is already defined on this. Hence, we have something like the following picture, depicting morphisms of $\mathcal{L}$ using $\rightarrowtail$, and morphisms of $\mathcal{R}$ using $\twoheadrightarrow$:


Now take a colimit of $\widehat{X} : J \downarrow i \to \mathcal{L}$, which is also a colimit in $\mathbf{C}$. For every $j < i$, we have a morphism $\widehat{X} j \xrightarrow{\varepsilon_j} X j \to X i$, so by the universal property, there is a unique morphism $p : \mathrm{colim}_{J \downarrow i} \widehat{X} \to X i$, as follows:
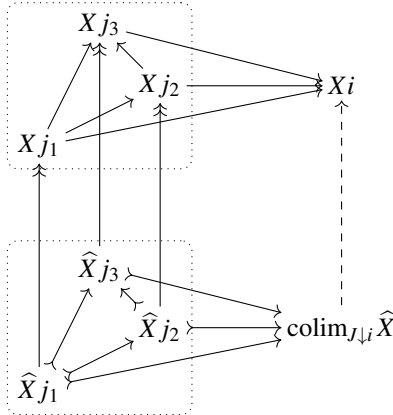
Now factorize the morphism p as $\operatorname{colim}_{J \downarrow i} \widehat{X} \xrightarrow{Lp} Fp \xrightarrow{Rp} Xi$.

Define $\widehat{X}i := Ep$. For every $j < i$, define $\widehat{X}j \to \widehat{X}i$ to be the composite $\widehat{X}j \to \operatorname{colim}_{J \downarrow i} \xrightarrow{Lp} \widehat{X}i$ which is in $\mathscr{L}$ because the colimit inclusion is in $\mathscr{L}$. Finally, define $\varepsilon_i := Rp$ (naturality is trivial to check).
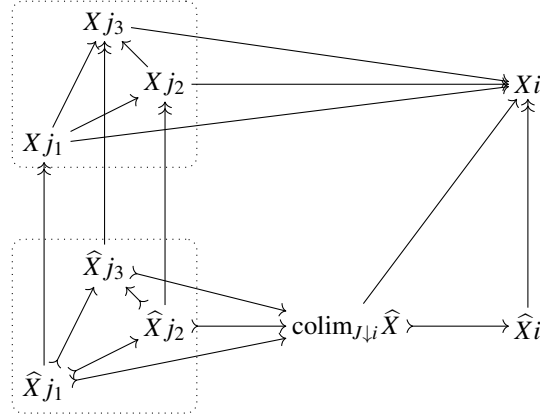
To prove functoriality, we proceed by induction again. For any minimal $i \in J$, let $\widehat{\alpha}_i := \alpha_i$. Now for any non-minimal $i \in J$, we have a natural transformation $\widehat{f} : \widehat{X} \Rightarrow \widehat{Y}$ on $J \downarrow i$ by induction. By the universal property, this induces a morphism between the colimits such that the following diagram commutes:

$$
\begin{array}{ccc}
\operatorname{colim}_{J \downarrow i} \widehat{X} & \xrightarrow{p_X} & Xi \\
\downarrow & & \downarrow{\scriptstyle \alpha_i} \\
\operatorname{colim}_{J \downarrow i} \widehat{Y} & \xrightarrow[p_Y]{} & Yi
\end{array}
$$

Hence, since the factorization is functorial, we get a morphism α_i with the required commutative square:

$$
\begin{array}{ccc}
\widehat{X}i & \xrightarrow{\varepsilon_{X,i}} & Xi \\
{\scriptstyle \widehat{\alpha}_i}\downarrow & & \downarrow{\scriptstyle \alpha_i} \\
\widehat{Y}i & \xrightarrow[\varepsilon_{Y,i}]{} & Yi
\end{array}
$$

$\square$

4 Layout algorithm

We begin by defining the layout problem in more precise terms, using the explosion construction.

Definition 4.1. A *layout* for an n-diagram X is a graph embedding of $\mathsf{pt}^n(X)$ into $\mathbb{R}^n$.

We are primarily interested in layouts that produce aesthetically pleasing string diagrams, since this forms an important part of homotopy.io where users can only view and interact with terms graphically. The goal of this section is to describe an algorithm that computes such layouts for arbitrary n-diagrams.

4.1 Layout for zigzags

First, let us consider the easier problem of layout for zigzags or 1-diagrams.

Definition 4.2. A *layout* for a zigzag X consists of:

- a real number $w \in \mathbb{R}$ called the *width*, together with

- an injective monotone map $\lambda : \mathbf{n} \to [0, w]$ where n is the length of X.

This data can be extended to a map assigning positions to all objects of the zigzag, by interpolating the positions of the regular objects (except for the first and last). We define $\lambda' : \mathsf{pt}(X) \to \mathbb{R}$ as follows:

$$\lambda'(\mathsf{s}i) := \lambda(i) + 1 \qquad \lambda'(\mathsf{r}j) := \begin{cases} 0 & \text{if } j = 0 \\ \frac{1}{2}(\lambda(j-1) + \lambda(j)) + 1 & \text{if } 0 < j < n \\ w + 2 & \text{if } j = n \end{cases} \tag{4}$$

If the zigzag has no singular objects, we set $\lambda'(\mathsf{r}0) := \frac{1}{2}w + 1$ which is the midpoint of $[0, w+2]$.

$$\bullet \longrightarrow 0 \longleftarrow \bullet \longrightarrow 1 \longleftarrow \bullet \qquad \rightsquigarrow \qquad 0 \longrightarrow 1 \longleftarrow \tfrac{3}{2} \longrightarrow 2 \longleftarrow 3$$

Figure 3: A layout for a zigzag.

More generally, we will want to compute a layout for a family of zigzags subject to certain aesthetic conditions relating the layouts of each zigzag according to some zigzag maps between them.

Definition 4.3. Given a functor $X : \mathbf{J} \to \mathrm{Zig}(\mathbf{C})$, a *layout* consists of:

- a real number $w \in \mathbb{R}$ called the *width*, together with

- a family of injective monotone maps $\lambda_i : (Xi)_\mathsf{s} \to [0, w]$ for every $i \in \mathbf{J}$.

Given a (non-identity) morphism $f : i \to j$ in $\mathbf{J}$, we say that the layout is *f-fair* if

$$\forall q \in (Xj)_\mathsf{s}. \, \lambda_j(q) \text{ is the average of } \{\lambda_i(p) \mid p \in (Xi)_\mathsf{s} \text{ such that } (Xf)_\mathsf{s}(p) = q\}.$$

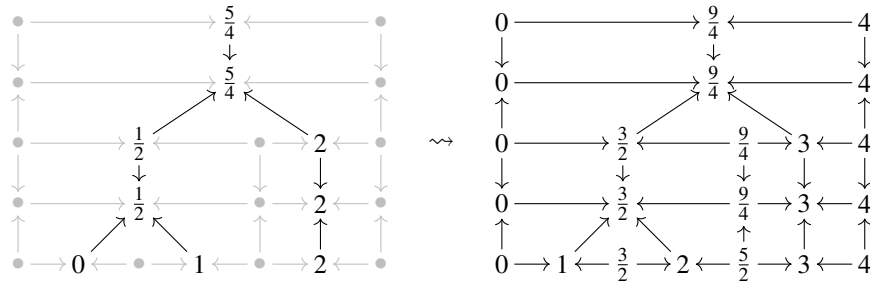If a layout is fair for every (non-identity) morphism, then we simply say that the layout is *fair*.

Figure 4: A fair layout for an exploded 2-diagram.

We are primarily interested in functors of the form $\mathsf{pt}^n(X) \to \mathrm{Zig}(\mathbf{C})$ which are the *n-fold explosion* of an $(n+1)$-diagram X. A layout for such a functor can be extended, using (4), to a map $\mathsf{pt}^{n+1}(X) \to \mathbb{R}$. This does not give a full layout for X, but it is like assigning x-coordinates to every point of X.

The fairness condition ensures that points are centred with respect to their inputs and outputs, and in particular, that identities are rendered as straight wires or flat surfaces. In general, it is not possible to get a fair layout for $\mathsf{pt}^n(X) \to \mathrm{Zig}(\mathbf{C})$ for $n \geq 2$. For example, in the 3-diagram in Figure 5, it is not possible to assign x-coordinates in a way that makes the top surface flat in both the y- and z-axes. However, what we are looking for is a layout that is as close to fair as possible, which becomes an optimisation problem.
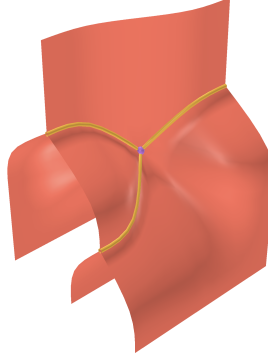
Figure 5: The associator 3-diagram as seen in homotopy.io.

Zigzag layout algorithm.   Given a finite poset-shaped diagram

$$X : J \to \mathrm{Zig}(\mathbf{C})$$

we obtain a (fair) layout according to the following scheme:

(1) Build the diagram $J \xrightarrow{X} \mathrm{Zig}(\mathbf{C}) \to \Delta \hookrightarrow \mathbf{Pos}$ which we denote by P.

(2) Obtain its mono-epi injectification according to Theorem 3.9:

$$
\begin{array}{ccc}
J & \xrightarrow{\quad P \quad} & \mathbf{Pos} \\
 & \widehat{P} \searrow \quad \Uparrow \varepsilon \quad \nearrow & \\
 & \mathbf{Pos}_{\mathrm{inj}} &
\end{array}
$$

Assume wlog that $Pi \subseteq \widehat{P}i$ for all $i \in J$ and $\widehat{P}i \subseteq \widehat{P}j$ whenever $i < j$.

(3) Construct a linear programming problem using the following scheme:[3]

$$
\begin{aligned}
\textbf{minimise} \quad & w + \sum m_{i,j,p} \\
\textbf{subject to} \quad & 0 \le v_{i,p} \le w & (i \in J,\, p \in \widehat{P}i) \quad (\star) \\
& v_{i,q} - v_{i,p} \ge 1 & (p < q \text{ in } \widehat{P}i) \quad (*) \\
& v_{i,q} - v_{i,p} \ge 1 & (\varepsilon_i(p) < \varepsilon_j(q) \text{ in } Pi) \quad (\diamond) \\
& |v_{i,p} - v_{j,p}| \le m_{i,j,p} & (i < j) \quad (\dagger) \\
& v_{j,q} = \mathrm{avg}(\{v_{j,p} \mid p \in Pi,\, P_{i<j}(p) = q\}) & (i < j,\, q \in Pj) \quad (\ddagger)
\end{aligned}
$$

(4) Extract the layout from the solution: $\lambda_i(p) := v_{i,p}$ for $i \in J$ and $p \in Pi$.

The idea is as follows. The set $\widehat{P}i$ is a disjoint union of all Pj for $j \le i$. Therefore, we have a variable $v_{i,p}$ for every $p \in Pj$ for $j \le i$. We think of this as the value of $\lambda_j(p)$ locally at i, with the true value being the one at j. This gives for each i a *local layout* for the points in the neighbourhood of i. The constraints $(\dagger)$ assert that the local layouts agree on common points, whereas the constraints $(\ddagger)$ impose that each local layout is fair, and together, they imply that the global layout is fair. Of course, this would not be satisfiable, so the constraints $(\dagger)$ are weakened so that the difference between local layouts is minimised instead of made zero. This gives a layout that, while not fair, is as fair as possible.

---

[3]In the implementation, we solve this using HiGHS which is an open-source linear programming solver [8].

4.2   Layout for diagrams

We can obtain a layout for any n-diagram X recursively:

(1) If X is a 0-diagram, its layout is trivial:

$$\lambda : \mathsf{pt}^0(X) \to \mathbb{R}^0$$
$$[] \mapsto ()$$

(2) If X is an $(n+1)$-diagram, compute its layout as an n-diagram:

$$\lambda_{\mathrm{ind}} : \mathsf{pt}^n(X) \to \mathbb{R}^n$$

(3) Construct its n-fold explosion $\mathsf{pt}^n(X) \to \mathsf{Zig}(\mathbf{C})$.

(4) Obtain its layout using the zigzag layout algorithm:

$$\lambda_{\mathrm{zig}} : \mathsf{pt}^{n+1}(X) \to \mathbb{R}$$

(5) Finally, combine this with the inductive layout to get a full layout for X:

$$\lambda : \mathsf{pt}^{n+1}(X) \to \mathbb{R}^{n+1}$$
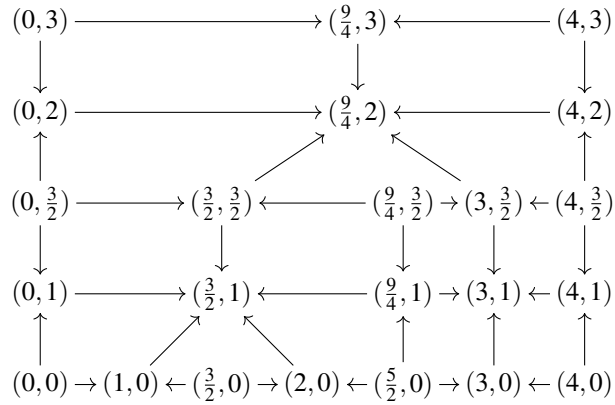$$p \mapsto (\lambda_{\mathrm{zig}}(p), \lambda_{\mathrm{ind}}(\mathrm{tail}\, p))$$

Figure 6: A full layout for a 2-diagram (combining the layouts of Figures 3 and 4).

References

[1] Krzysztof Bar & Jamie Vicary (2017): *Data structures for quasistrict higher categories*. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, IEEE, pp. 1–12, doi:10.1109/LICS.2017.8005147.

[2] Simon Burton (2023): *String diagrams for higher mathematics with wiggle.py*. Talk at SYCO 11. Available at `https://www.cl.cam.ac.uk/events/syco/11/slides/burton.pdf`.

[3] Nathan Corbyn, Lukas Heidemann, Nick Hu, Chiara Sarti, Calin Tataru & Jamie Vicary (2023): *The proof assistant homotopy.io*. Available at `https://homotopy.io`.

[4] Christoph Dorn (2018): *Associative n-categories*. Ph.D. thesis, University of Oxford.

[5] Christoph Dorn, Christopher Douglas & Jamie Vicary (2023): *The theory of associative n-categories*. In preparation.

[6] Lukas Heidemann (2023): *Framed combinatorial topology with labels in ∞-categories*. In preparation.

[7] Lukas Heidemann, David Reutter & Jamie Vicary (2022): *Zigzag normalisation for associative n-categories*. In: *37th Annual ACM/IEEE Symposium on Logic in Computer Science*, ACM, pp. 1–13, doi:10.1145/3531130.3533352.

[8] Q. Huangfu & J. A. J. Hall (2018): *Parallelizing the dual revised simplex method*. Mathematical Programming Computation 10(1), pp. 119–142, doi:10.1007/s12532-017-0130-5.

[9] Aleks Kissinger & Vladimir Zamdzhiev (2015): *Quantomatic: a proof assistant for diagrammatic reasoning*. In: *25th International Conference on Automated Deduction*, Springer, pp. 326–336, doi:10.1007/978-3-319-21401-6_22.

[10] David Reutter & Jamie Vicary (2019): *High-level methods for homotopy construction in associative n-categories*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science*, IEEE, pp. 1–13, doi:10.1109/LICS.2019.8785895.

[11] Emily Riehl (2008): *Factorization systems*. Notes from a talk. Available at `https://emilyriehl.github.io/files/factorization.pdf`.

[12] Paweł Sobociński, Paul W Wilson & Fabio Zanasi (2019): *CARTOGRAPHER: a tool for string diagrammatic reasoning*. In: *8th Conference on Algebra and Coalgebra in Computer Science*, pp. 1–7, doi:10.4230/LIPIcs.CALCO.2019.20.

[13] Gavin C Wraith (1993): *Using the generic interval*. Cahiers de topologie et géométrie différentielle catégoriques 34(4), pp. 259–266.

A   Universal zigzag bundle

The zigzag construction can also be described as a polynomial functor. This is due to Heidemann [6] and can be used to give an elegant defintion of explosion as a left adjoint of the zigzag construction.

Definition A.1. The *universal zigzag category* $\mathbf{Z}$ is the category with *singular objects* $(n,\mathsf{s}i)$ for $n \in \mathbb{N}$ and $0 \leq i < n$, *regular objects* $(n,\mathsf{r}j)$ for $n \in \mathbb{N}$ and $0 \leq j \leq n$, and morphisms given by:

$$
\begin{aligned}
\mathbf{Z}((n,\mathsf{s}i),(m,\mathsf{s}j)) &= \{f : \mathbf{n} \to \mathbf{m} \mid f(i) = j\} \\
\mathbf{Z}((n,\mathsf{r}i),(m,\mathsf{r}j)) &= \{f : \mathbf{n} \to \mathbf{m} \mid \mathsf{R}f(j) = i\} \\
\mathbf{Z}((n,\mathsf{r}i),(m,\mathsf{s}j)) &= \{f : \mathbf{n} \to \mathbf{m} \mid \mathsf{R}f(j) \leq i \leq \mathsf{R}f(j+1)\} \\
\mathbf{Z}((n,\mathsf{s}i),(m,\mathsf{r}j)) &= \varnothing
\end{aligned}
$$

There is a canonical forgetful functor, called the *universal zigzag bundle*,

$$
\begin{array}{c}
\mathbf{Z} \\
\downarrow{\scriptstyle p} \\
\Delta
\end{array}
$$

sending both $(n,\mathsf{s}i)$ and $(n,\mathsf{r}j)$. Now consider diagrams of the following form in **Cat**:

$$
\begin{array}{ccc}
\mathbf{C} \longleftarrow & \mathbf{J} \times_\Delta \mathbf{Z} & \longrightarrow \mathbf{Z} \\
& \downarrow \qquad \lrcorner & \downarrow{\scriptstyle p} \\
& \mathbf{J} \longrightarrow & \Delta
\end{array}
$$

If we let $\mathbf{J} = \mathbf{1}$, then $\mathbf{1} \to \Delta$ picks an object $n \in \mathbb{N}$. Then the fiber $\mathbf{Z}_n := \mathbf{1} \times_\Delta \mathbf{Z}$ is a category representing the "shape" of a zigzag of length n, and a functor $\mathbf{Z}_n \to \mathbf{C}$ is precisely a zigzag of length n in $\mathbf{C}$.

Proposition A.2. *The functor p is exponentiable, i.e. the pullback functor p^* has a right adjoint:*

$$
\mathbf{Cat}/\Delta \quad \overset{p^*}{\underset{\Pi_p}{\rightleftarrows}} \perp \quad \mathbf{Cat}/\mathbf{Z}
$$

We can now recover the zigzag construction from (2) as the polynomial functor

$$
\mathbf{Cat} \xrightarrow{\;!^*\;} \mathbf{Cat}/\mathbf{Z} \xrightarrow{\;\Pi_p\;} \mathbf{Cat}/\Delta \xrightarrow{\;\Sigma_!\;} \mathbf{Cat}
$$

where $\Sigma_!$ is the the left adjoint of $!_*$ (which always exists). In particular, we have an adjunction

$$
\mathbf{Cat}/\Delta(\mathbf{J}, \mathrm{Zig}(\mathbf{C})) \cong \mathbf{Cat}(\mathbf{J} \times_\Delta \mathbf{Z}, \mathbf{C}).
$$

which is precisely the explosion construction; given $X : \mathbf{J} \to \mathrm{Zig}(\mathbf{C})$, we have that $\mathrm{pt}(X) \cong \mathbf{J} \times_\Delta \mathbf{Z}$.