Representing and Enumerating Two-Dimensional Pasting Diagrams

James Cranch

October 15, 2015

1 Introduction

The study of category theory leads naturally to the study of 2-*categories*, a concept where there are objects (often depicted graphically as vertices, and often correspondingly called 0-*cells*), morphisms between those objects (thought of as directed edges, and correspondingly called 1-*cells*) and morphisms between those morphisms (thought of as faces, and correspondingly called 2-*cells*).

The motivating example of a 2-category has 0-cells given by categories, 1-cells given by functors and 2-cells given by natural transformations. In this example, and in general, the 1-cells and 2-cells can be composed in various ways, subject to various axioms.

One is naturally enough drawn to consider the free 2-category on some generating 0-cells, 1-cells and 2-cells. A 2-cell in this free 2-category is called a 2-*dimensional pasting diagram*, or in this paper a 2-*diagram* for short.

It can be represented as a directed acyclic graph G with a unique source and a unique sink, together with an embedding of G into $\mathbb{R}^2$ (considered up to isotopy) such that the source and sink both lie on the outside of the diagram. The (interior) faces of the resulting planar graph are the 2-cells.

For the sake of description, will choose to draw all our 2-diagrams with the source on the left and the sink on the right, and all edges pointing rightwards. This convention is normal in category theory (in graph theory, on the other hand, edges usually point up). We will use geographical language which corresponds to this particular convention.

It is natural to seek an inductive characterisation of such planar graphs, and hence of 2-cells in a free 2-category. The 1-dimensional analogue is straightforward: a 1-dimensional pasting diagram is simply a chain of arrows, and the appropriate structure for storing such a thing is the linked list: it is either empty, or consists of an arrow followed by a linked list of arrows.

How do we produce a 2-dimensional version? We need to make a choice of the first 2-cell to consider. Even in the 1-dimensional case, we could have represented a chain of morphisms as a linked list in two different ways, by starting at either end, and there are even more choices available here.

Our choice is to start by describing first the topmost 2-cell incident with the source vertex; let this be called x. On the left of Figure 1 is a particular 2-diagram with x labelled.

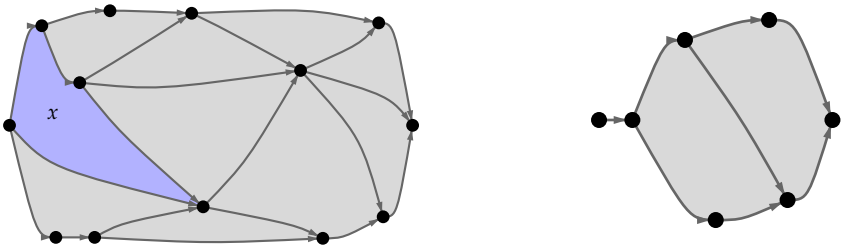

Figure 1 – On the left, an example 2-diagram with first cell x; on the right, an example 2-diagram with delayed first cell.

Of course, as shown on the right of Figure 1, there may in fact be no 2-cell incident with the source. In this case we record that the source has a single outgoing edge, and recursively describe the 2-diagram that we find at the far end of it. (Indeed, there may not even be an outgoing edge, and in that case we record that the diagram is merely a point).

In the case where there is such an x, the problem is how to describe what remains. There may very well be other cells above it, so long as they start after the first edge of the top boundary of x, and there may very well be other cells below this first cell, subject to no such restriction. It is natural to recognise that these two regions comprise 2-diagrams themselves, and natural to wish to choose them so that they partition the rest of the diagram.

The problem is that this cannot be done in a unique way. In the particular example considered above, there are several different ways of partitioning the whole 2-diagram into the single cell x and two other diagrams A (above x) and B (below x): they differ in terms of which of A and B gets each of the three cells in the bottom right. Figure 2 depicts them.

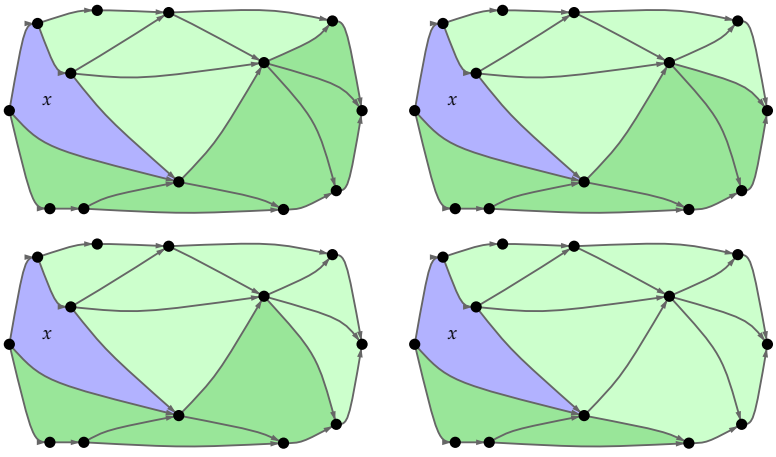

Figure 2 – The four ways of choosing A and B.

Our choice shall always be the one where B gets as much as possible and A gets as little as possible. In our example, this is the choice depicted on the top left in Figure 2.

To accommodate this, we need to extend the definition of 2-diagram a little, in order to be able to legislate that A may not contain any cells that should properly belong to B. Our technique for doing this — the sole modest innovation introduced in this document — is the notion of a *shaved* 2*-diagram*.

A shaved 2-diagram has its 1-target expressed as the concatenation of two parts, the *unshaved section* and the *shaved section*, with the property that which no 2-cells may have their source on the shaved section. (We think of 2-cells growing along the boundary as being like hair, and the term "shaved" is intended to connote that care has been taken to prevent the emergence of hair). We refer to this extra structure as a *shaving*. We should be clear that a shaving is not a property of a 2-diagram, but is genuinely a structure upon it: the same 2-diagram can often be given several different shavings.

In our illustrations we show the shaved section with visible stubble, as in Figure 3.

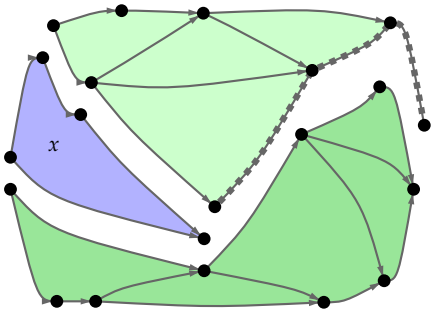

Figure 3 – Our example diagram, shown dissected into x, A and B, with the shaved portion of A shown.

Using this technique, we can inductively characterise shaved 2-diagrams as being one of the following:

(0) a 1-dimensional chain of edges, all shaved;

(1) a single unshaved edge with a shaved 2-diagram on the right of it; or

(2) a 2-cell x, with shaved 2-diagrams A above and B below as described earlier, such that A's shaved section is precisely its common boundary with B.

In case (2), the shaved and unshaved sections of the resulting 2-diagram are those of B. Hence we must demand that the unshaved section of B is nonempty (since if it were all shaved, that would be incompatible with the presence of x).

Then we recover the ordinary (unshaved) notion of 2-diagrams by looking at shaved 2-diagrams with empty shaved section.

This paper is only part of a project; another part consists of some computer programs to manipulate the data structures described (these were used, for example, to generate all the diagrams). The homepage for this project is:

`http://jdc41.user.srcf.net/research/pasting/`

2 Formal definitions

For a basic introduction to 2-categories, see [7], and for a general introduction to the language and graphical concepts of higher categories, see [6]. We will use the language and imagery extensively: though we provide translations into combinatorial language and an implementation in dependent type theory, the reader will miss half of the motivation this way.

A 2-*diagram* for short, is a 2-cell of a free strict 2-category on a given set of 0-cells C_0, a given set of generating 1-cells $C_1(x, y)$ between them, and a given set of generating 2-cells $C_2(\alpha, \beta)$ between composites of those. Unless we state otherwise, the generating 2-cells will all be *positive*: their source and target 1-cells are not the empty composite. Later we will consider the *half-positive* case where the target 1-diagram can be empty but the source 1-diagram still may not be.

We write 1-*source* and 1-*target* to mean the source and target 1-diagram; the 0-*source* and 0-*target* are the source and target 0-diagram.

According to the standard graphical calculus for strict 2-categories, these are in bijection with directed acyclic graphs G with a unique source and a unique sink, together with an embedding of G into $\mathbb{R}^2$ (considered up to isotopy) such that the source and sink both lie on the boundary, where every vertex is labelled with an element of C_0, every edge with source x and target y is labelled with an element of $C_1(x, y)$, and every interior face is labelled by an element of $C_2(\alpha, \beta)$, where α is the upper and β the lower boundary. Here, as in the introduction, we are imagining the graph embedded so that the source is on the left and the sink is on the right, and that all 1-cells point rightwards, and all 2-cells point downwards. This convention is standard in 2-category theory.

Given a 2-diagram A with 1-source α and 1-target β, a *shaving* of A is a decomposition $\beta = \sigma\upsilon$ (into an *unshaved* and a *shaved* section) such that if A is written as $A = B \odot (C * 1_\upsilon)$, where $*$ denotes horizontal and $\odot$ vertical composition, as depicted in Figure 10, then $C = 1_\sigma$.

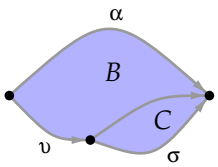

Figure 4 – A decomposition of a shaving diagram forcing $C = 1_\sigma$.

Graphically, this says that any vertex which has a shaved edge emanating from it has only that edge emanating from it.

Remark 1. Any 2-diagram can be given the *empty* shaving: ordinary 2-diagrams are the same thing as shaved 2-diagrams with empty shaved section.

Remark 2. If a 2-diagram is *fully shaved* (which means that the shaved section consists of the entirety of the 1-target) then that 2-diagram has no 2-cells.

We can give an inductive definition of shaved 2-diagrams within dependent type theory. Indeed, we suppose the following prerequisite (dependent) types, whose elements represent the 0-cells and generating 1-cells of our free strict

2-category:

$$\frac{}{\mathrm{C}_0\colon \mathrm{Type}}, \qquad \frac{x, y\colon \mathrm{C}_0}{\mathrm{C}_1(x,y)\colon \mathrm{Type}}.$$

This allows us to form a dependent type of 1-diagrams and constructors thus:

$$\frac{x, y\colon \mathrm{C}_0}{\mathrm{D}_1(x,y)\colon \mathrm{Type}},$$

$$\frac{x\colon \mathrm{C}_0}{\mathrm{nil}\colon \mathrm{D}_1(x,x)},$$

$$\frac{x, y, z\colon \mathrm{C}_0 \quad h\colon \mathrm{C}_1(x,y) \quad \tau\colon \mathrm{D}_1(y,z)}{\mathrm{cons}(h,\tau)\colon \mathrm{D}_1(x,z)}.$$

Notice that we omit some arguments from the constructors: this shall be standard practice for us, and it is to be understood that we implicitly claim that these can always be deduced from context.

We will also need a recursive function

$$\frac{x, y, z\colon \mathrm{C}_0}{\mathrm{append}\colon \mathrm{D}_1(x,y) \to \mathrm{D}_1(y,z) \to \mathrm{D}_1(x,z)},$$

defined by

$$\mathrm{append}(\mathrm{nil}, \beta) = \alpha$$

$$\mathrm{append}(\mathrm{cons}(h, \alpha), \beta) = \mathrm{cons}(h, \mathrm{append}(\alpha, \beta)).$$

Given, then, some dependent type of generating 2-cells:

$$\frac{\begin{array}{c} x, y, y', z\colon \mathrm{C}_0 \quad h\colon \mathrm{C}_1(x,y) \quad h'\colon \mathrm{C}_1(x,y') \\ \tau\colon \mathrm{D}_1(y,z) \quad \tau'\colon \mathrm{D}_1(y',z) \end{array}}{\mathrm{C}_2(\mathrm{cons}(h,\tau), \mathrm{cons}(h',\tau'))\colon \mathrm{Type}},$$

we can define the type of shaved 2-diagrams and its constructors as follows:

$$\frac{x, y, z\colon \mathrm{C}_0 \quad \alpha\colon \mathrm{D}_1(x,z) \quad \beta\colon \mathrm{D}_1(x,y) \quad \gamma\colon \mathrm{D}_1(y,z)}{\hat{\mathrm{D}}_2(\alpha,\beta,\gamma)\colon \mathrm{Type}}$$

$$\frac{x, y\colon \mathrm{C}_0 \quad \alpha\colon \mathrm{D}_1(x,y)}{\mathrm{sd}_0(\alpha)\colon \hat{\mathrm{D}}_2(\alpha, \mathrm{nil}, \alpha)}$$

$$\frac{\begin{array}{c} w, x, y, z\colon \mathrm{C}_0 \quad h\colon \mathrm{C}_1(w,x) \quad \alpha\colon \mathrm{D}_1(x,z) \\ \beta\colon \mathrm{D}_1(x,y) \quad \gamma\colon \mathrm{D}_1(y,z) \quad A\colon \hat{\mathrm{D}}_2(\alpha,\beta,\gamma) \end{array}}{\mathrm{sd}_1(h,A)\colon \hat{\mathrm{D}}_2(\mathrm{cons}(h,\alpha), \mathrm{cons}(h,\beta), \gamma)}$$

$$\frac{\begin{array}{c} w, x, x', x'', y, y', z\colon \mathrm{C}_0 \quad h\colon \mathrm{C}_1(w,x) \quad h'\colon \mathrm{C}_1(w,x') \quad h''\colon \mathrm{C}_1(w,x'') \\ \alpha\colon \mathrm{D}_1(x,z) \quad \beta\colon \mathrm{D}_1(x,y) \quad \gamma\colon \mathrm{D}_1(y,z) \\ \beta'\colon \mathrm{D}_1(x',y) \quad \beta''\colon \mathrm{D}_1(x'',y') \quad \gamma'\colon \mathrm{D}_1(y',z) \\ \theta\colon \mathrm{C}_2(\mathrm{cons}(h,\beta), \mathrm{cons}(h',\beta')) \quad A\colon \hat{\mathrm{D}}_2(\alpha,\beta,\gamma) \\ B\colon \hat{\mathrm{D}}_2(\mathrm{cons}(h', \mathrm{append}(\beta',\gamma)), \mathrm{cons}(h'',\beta''), \gamma') \end{array}}{\mathrm{sd}_2(\theta, A, B)\colon \hat{\mathrm{D}}_2(\mathrm{cons}(h,\alpha), \mathrm{cons}(h'',\beta''), \gamma')}.$$

The reader can verify that these three constructors make formal the intuitive descriptions given in the introduction: sd_0, sd_1, and sd_2 respectively realise cases (0), (1) and (2) as described above.

Now, finally, we can introduce ordinary 2-diagrams as a special case:

$$\frac{x, y\colon \mathrm{C}_0 \quad \alpha, \beta\colon \mathrm{D}_1(x, y)}{\mathrm{D}_2(a, b)\colon \mathrm{Type}},$$

$$\frac{x, y\colon \mathrm{C}_0 \quad \alpha, \beta\colon \mathrm{D}_1(x, y) \quad d\colon \hat{\mathrm{D}}_2(\alpha, \beta, \mathrm{nil})}{|d|\colon \mathrm{D}_2(\alpha, \beta)}.$$

3 Structure on 2-diagrams

Sources and targets

It is quite likely that practical implementors will not wish to store data about sources and targets in each shaved 2-diagram (given the recursive nature of the definition, that would be a considerable amount of data). Hence it is worth studying how it can be recovered from the rest of the data.

In practice, most applications will see it possible to deduce the source and target 1-diagrams of each component 2-cell. However, when that is not the case, all that need be stored is the initial 1-cell of the 1-source of each cell: we write $\mathrm{hs}_1(\theta)$ for this in the formulae below.

Given that, the source and target 1-diagrams of a shaved 2-diagram may be defined recursively as follows:

$$\begin{aligned}
\mathrm{source}_1(\mathrm{sd}_0(\alpha)) &= \alpha, \\
\mathrm{source}_1(\mathrm{sd}_1(h, A)) &= \mathrm{cons}(h, \mathrm{source}_1(A)), \\
\mathrm{source}_1(\mathrm{sd}_2(\theta, A, B)) &= \mathrm{cons}(\mathrm{hs}_1(\theta)), \mathrm{source}_1(A)); \\
\mathrm{target}_1(\mathrm{sd}_0(\alpha)) &= \alpha, \\
\mathrm{target}_1(\mathrm{sd}_1(h, A)) &= \mathrm{cons}(h, \mathrm{target}_1(A)), \\
\mathrm{target}_1(\mathrm{sd}_2(\theta, A, B)) &= \mathrm{target}_1(B).
\end{aligned}$$

Horizontal composition

Horizontal composition of 2-diagrams (which, of course, is defined to correspond with horizontal composition as 2-cells) is easy to define by a recursive function. We write the horizontal composition of A and B, with A on the left and B on the right, as $B \star A$. This convention is geometrically annoying but it agrees with the standard backwards notation for functional composition.

We demand that shavings match: that is, for $B \star A$ to make sense, if A has nonempty shaved portion, then B is fully shaved.

Given that, we can define

$$B \star \mathrm{sd}_0(\mathrm{nil}) = B$$

$$\mathrm{sd}_0(\beta) \star \mathrm{sd}_0(\alpha) = \mathrm{sd}_0(\beta \circ \alpha)$$

$$B \star \mathrm{sd}_1(h, A) = \mathrm{sd}_1(h, B \star A)$$

$$C \star \mathrm{sd}_2(\theta, A, B) = \mathrm{sd}_2(\theta, \mathrm{sd}_0(\mathrm{source}_1\, C) \star A, C \star B).$$

Vertical composition

Vertical composition of 2-diagrams, written $A \odot B$, where B is above A, is a little trickier to define. If we define it by structural recursion on the two diagrams involved, all cases are clear except when we are gluing something given by sd_1 above to something given by sd_2 below.

This case is difficult, because the result will end up being an sd_2 like the lower part, and the sd_1 above will have to be cut apart to fit it into the structure.

A useful notion here is that of the *right transparency* of a shaved 2-diagram A: this is the largest 1-diagram α such that $A = \mathrm{sd}_0(\alpha) \star A'$. Clearly this is less than the length of the shaved section. The right transparency can be defined recursively as follows:

$$\mathrm{rtrans}(\mathrm{sd}_0(\alpha)) = \alpha$$

$$\mathrm{rtrans}(\mathrm{sd}_1(h, A)) = \mathrm{rtrans}(A)$$

$$\mathrm{rtrans}(\mathrm{sd}_2(\theta, A, B)) = \mathrm{rtrans}(A) \cap \mathrm{rtrans}(B)$$

where $\cap$ denotes the evident (and easily-defined) intersection function.

The significance of the right transparency is that it tells us where to cut the upper diagram apart, and it is not difficult to recursively define a function which cuts accordingly, and having that it is straightforward to define the vertical join.

Whiskering

For comparison with other formalisms, it may be desirable to convert the 2-diagram to whiskered form: that is, to write it as a vertical join of cells horizontally joined with thin diagrams.

This is a straightforward recursive function.

Pasting

Pasting could be regarded as a more fundamental algebraic structure than joins. Indeed, vertical and horizontal joins can be implemented in terms of pastings indexed by diagrams of two-cells.

However, for simplicity's sake we advocate treating joins as fundamental and implementing pastings in terms of them.

Indeed, suppose we have a 2-diagram A, and for each i-cell ($i = 0, 1, 2$) of A an i-diagram B_θ, compatible in the obvious sense, and we wish to produce the pasting of the B's according to A. We can do this without ceremony by recursing through the structure of A and making vertical joins as appropriate.

4 Enumeration strategies

In this section we review strategies for various sorts of enumeration problem using data of the above sort. Several examples are provided in the next section.

4.1 Specifying source and target

Suppose given a free 2-category: we could attempt to enumerate all 2-diagrams in it with a specified 1-source and 1-target.

Our suggested approach is to generalise, and produce an algorithm for enumerating 2-diagrams with specified 1-source, specified unshaved section of the 1-target. There may be several possible shaved sections of the 1-target, and we wish to enumerate then indexed according to these possibilities.

This can now readily be done by a straightforward recursive method: for each 1-source and unshaved 1-target section, we loop through all possible initial 2-cells x (this data can be cached in advance). For each one, the 2-diagram above it has known 1-source (the given 1-source, less its initial 1-cell), and known unshaved section of its 1-target (the 1-source of our 2-cell, less its initial 1-cell). We recurse to enumerate through diagrams of this shape: each different possible shaved 1-target section gives us (together with the 1-target of x) a complete 1-section for the diagram below x.

4.2 Specifying cells

One could instead count based on the number of cells (0-cells, 1-cells or 2-cells) in the diagram, or perhaps the numbers of cells of each type. Naturally, there are some restrictions if the resulting number must be finite. For example, if we're restricting by the total number of 0-cells, we may have to be careful with our generating 2-cells with source and target of length 1: we do not want to be able to form arbitrarily long vertical joins of such cells, as that gives us an infinite number of diagrams with just two 0-cells.

However, once conventions have been fixed, this can be done by a similar recursive approach to the above: counting by initial 2-cell, then by the diagram above it, then by the diagram below it.

Since the approach is similar, it can be combined with the above, to enumerate diagrams with specified 1-source and 1-target, and constraints on the cells used.

4.3 A more general family

Both the strategies above can be regarded as instances of the same thing, namely counting preimages of a certain diagram along some 2-functor from our free 2-category to a 2-category of some fairly straightforward kind.

Indeed, if $\mathcal{C}$ is a free 2-category, then write $\mathcal{C}_{/2}$ for the 2-category with the same 0-cells and 1-cells, but a unique 2-cell between any pair of parallel 1-cells. There is an obvious 2-functor from $\mathcal{C}$ to $\mathcal{C}_{/2}$, and the preimage of any 2-cell is the set of 2-diagrams of $\mathcal{C}$ with the same source and target.

Similarly, if we have a map f (of sets) from the generating 2-cells of $\mathcal{C}$ to a commutative monoid A, then that map extends uniquely to a 2-functor $\mathcal{C} \to A$. Here A is regarded as a 2-category thanks to the Eckmann-Hilton argument:

we have a unique 0-cell, a unique 1-cell, and 2-cells given by A, with both compositions given by the monoidal operation. Then the preimage of an element x in A is the set of 2-diagrams D with the sum of $f(a)$ over all 2-cells of D equal to x.

A similar recursive algorithm could be produced for any such problem, wherever the target 2-category is well enough understood that, given the intended image, we can recurse over possible images of each part of a shaved diagram.

5 Examples of enumeration

Unsurprisingly, a wide range of combinatorial problems can be rephrased in terms of labelled planar graphs, or as ways of rewriting words, and so on. As a result, 2-diagrams interact with several interesting questions in enumerative combinatorics.

The examples given below are largely motivated by being illustrations of the power of our calculus of shaved diagrams in manipulating certain combinatorial structures. However, it is entirely possible that some will be of independent interest.

5.1 Unlabelled diagrams

Perhaps the most obvious and most natural is to enumerate *unlabelled* 2-diagrams. From a 2-categorical perspective, this is considering 2-cells in the free strict 2 category on one generating 0-cell, one generating 1-cell, and a generating 2-cell between any two positive composites of the generating 1-cell.

If we allow a unique 2-cell of any positive source and target length, we recover the number of Baxter permutations on n objects [1, series A001181]. This is a sequence that has been studied extensively in this geometric context: for example, the paper [3] surveys some enumerative aspects, and produces an inductive description of them which is different to ours; however, their description appears much less likely to give pleasant descriptions of more general labellings.

5.2 Unlabelled graphs

Graph theorists may find it more interesting to eliminate multiple edges from the resulting diagrams to obtain a proper graph, by withholding the generating 2-cell with source and target of length 1.

The graph drawing community refer to these as *planar st-graphs* [2, Section 4.2]. Figure 5 shows the numbers of these with n edges, for n up to 20: these were straightforward to calculate by recursion. Since an early draft of this paper was made available, this sequence has been placed online as [1, series A236408].

5.3 Trees

There is an evident relationship between the theory of pasting diagrams and the combinatorics of triangulated polygons. Indeed, if we take just one gener-

1	1	11	295297
2	1	12	1526427
3	3	13	8061879
4	9	14	43380351
5	33	15	237266225
6	131	16	1316536991
7	561	17	7399318871
8	2535	18	42065753191
9	11971	19	241628448517
10	58579	20	1400957386207

Figure 5 – The numbers of unlabelled diagrams without multiple edges, with n 1-cells, for $n \leq 20$.

ating 2-cell, with source of length two and target of length one, we obtain the central binomial coefficients $\binom{n}{\lfloor n/2 \rfloor}$ as the numbers of $(n+1)$-edge 2-diagrams.

5.4 Dominoes and other tilings

If we allow ourselves a single generating 0-cell, two generating 1-cells a and b, and two generating 2-cells $R : ba^2 \Rightarrow a^2b$ and $S : b^2a \Rightarrow ab^2$, then the number of 2-cells from $b^n a^m$ to $a^m b^n$ is the number of domino tilings of an $m \times n$ rectangle:

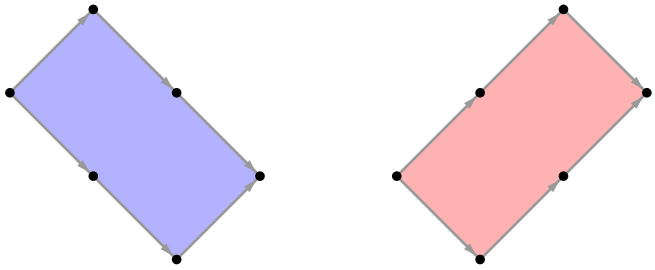

Figure 6 – Dominoes as 2-cells.

Other problems about dimer models on regular geometric configurations [5] (such as the triangular lattice) admit reformulations in this language.

Of course, better enumeration techniques exist for these problems, but the reformulation may still be of interest.

5.5 Nonconvex tiles

While the discussion above restricts itself to enumerating tilings where the tiles are convex (and hence which can be comprised of generating 2-cells in a 2-category whose 1-cells go from right to left), there need be no such restriction in general.

For example, consider tilings of plane regions by L-triominoes: connected unions of three squares which are not 3×1 rectangles.

According to the scheme described above, two will cause no problem, but the other two will not have a unique source or will not have a unique sink. This can be dealt with by splitting them up in a convenient fashion: we have chosen to divide them in two down the middle, so that both halves can be represented by 2-cells, and introducing two new 1-cells to represent the split edges thus revealed:

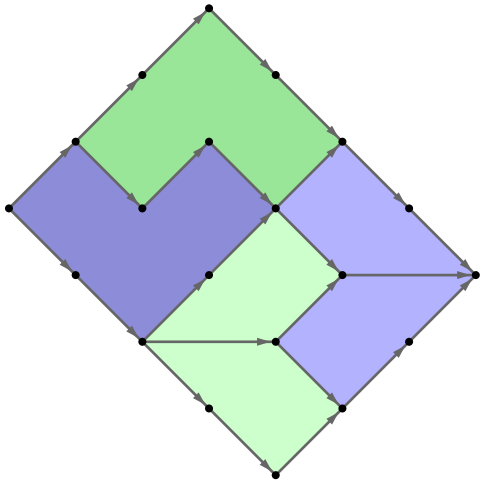

Figure 7 – L-triominoes

Suppose now that one wishes to enumerate instead tilings by triominoes. If we followed the above strategy, we would end up modelling the problem with a free 2-category exactly like the above, but with two extra generating 2-cells, representing the two orientations of the 3×1 rectangle.

There is however a smaller model: rather than adding extra morphisms, we can simplify by identifying the two extra 1-cells we added, so we just have one of them, and the four extra pentagons as before. This allows us to form the rectangles as follows:

Based on the existence of tricks like this, there seems to be no reason to think that finding optimal categorical representations of a tiling problem should be easy.

With this representation, and the methods described above, we were able to calculate that there are 5271923 tilings of a 6×8 rectangle with triominoes within a few moments. This method appears not to scale gracefully beyond this point for this particular brute-force enumeration task, but may be of use for certain specialist tasks (for example, it may scale better in situations where large parts of a tiling are forced, which is not the case here).

6 Extensions

6.1 Relation with Conway's tiling invariants

Conway introduced a group-theoretic invariant for tilings [4, 8], with an algebraic character similar in some ways to the present work; the reader might

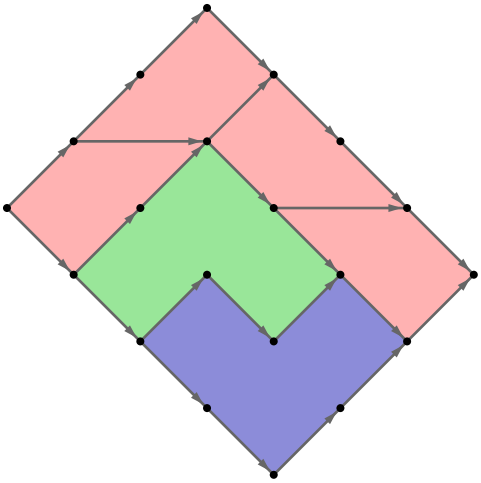

Figure 8 – 3×1 rectangles using the raw ingredients of L-tetrominoes

wonder what connection there may be.

Suppose given a free 2-category with a unique 0-cell, regarded as a tiling problem. Then the group associated to it by Conway is the quotient of the free group on the 1-cells by relations arising by equating the 1-source and 1-target of each 2-cell.

6.2 The half-positive case

Earlier, we promised the reader an account of the changes required to support half-positive 2-cells: those with possibly trivial 1-target (but nontrivial 1-source). Of course, the same technology could be used upside-down to allow 2-cells with possibly trivial 1-source but nontrivial 1-target instead.

The changes are straightforward.

The most obvious source of applications are those to word rewriting problems, where it is natural to allow rewriting rules which simply remove certain substrings entirely. In this context, half-positivity is natural: rewriting rules which add substrings from nowhere are usually undesirable, as they prevent rewriting systems from terminating.

The general non-positive case, where both sources and targets can be trivial, is combinatorially annoying: for example, one no longer expects the set of diagrams with specified source and target to be finite. As a result, we do not treat it.

6.3 General unions of 2-cells

The applications above to enumeration of tilings suggest a generalisation to storing more general graphs. From a category-theoretic perspective, we mean overlaps of 2-diagrams which are not necessarily themselves 2-diagrams; from a graph-theoretic perspective, we mean planar digraphs possibly with multiple sources and sinks.

In the simply-connected case, at least, it is possible to extend the techniques above to enumerate such diagrams with specified boundary.

The obvious approach to this is to introduce new generating cells allowing the diagram to be completed to a 2-diagram in a unique fashion. We may achieve this as follows: We adjoin two new 0-cells s and t to be the source and sink of our 2-diagram. Then, numbering the sources of our diagram $s_1, \ldots, s_m$ from top to bottom, we take a generating 1-cell f_i from s to s_i for each i; similarly we number the targets $t_1, \ldots, t_n$ and provide a generating 1-cell g_j from t_j to t for each j.

Then for each $i = 1, \ldots, m-1$ we add a generating 2-cell from $q_i f_i$ to $p_{i+1} f_{i+1}$, where p_i is the upper boundary emanating from s_i, and q_i is the lower boundary emanating from s_i.

Similarly, for each $j = 1, \ldots, n-1$ we add a generating 2-cell from $g_j q'_j$ to $g_{j+1} p'_{j+1}$, where p'_j is the upper boundary incoming into t_j, and q'_j is the lower boundary coming into t_j.

We then enumerate the diagrams with 1-source the genuine 1-source precomposed with f_1 and postcomposed with g_1, and 1-target the genuine 1-target precomposed with f_m and postcomposed with g_n.

This is depicted below, with the grey 2-cells the additional ones:

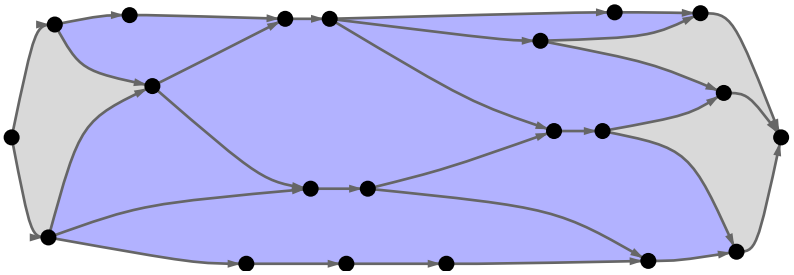

Figure 9 – A graph with multiple sources and sinks

In the non-simply connected case, things are easier: we may simply adjoin new generating 2-cell of the shapes desired as the holes, and count only 2-diagrams with precisely one occurrence of each such 2-cell.

6.4 The difficulties of higher dimensions

We aim now to persuade the reader that any comparable formalism for 3-categories will necessarily be rather more complicated.

Consider the free 2-category with nine objects, twelve generating 1-cells and four generating 2-cells A, B, C, D as below. By a slight abuse of notation, we write BD and CD for the obvious rectangular composites of those two pairs of cells. Now consider the free 3-category generated over this data by 3-cells $\Theta : BD \Rightarrow BD$ and $\Xi : CD \Rightarrow CD$.

Now consider an alternating composite $\Theta\Xi \cdots \Theta\Xi$. This is more annoying to represent than any 2-diagram, in some concrete sense. There is no 3-cell whose 2-source mentions A, but this does not allow us to defer discussion to the two-cells B and C one after the other: there are 3-cells using each in an alternating fashion. The author has found this a useful counterexample to several

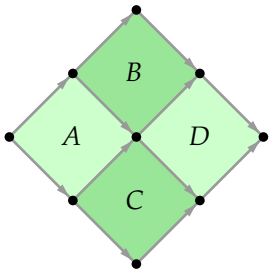

Figure 10 – A certain 2-diagram

naive attempts at a generalisation.

References

[1] Neil J. A. Sloane (ed.), *The Online Encyclopedia of Integer Sequences*. Online at `http://oeis.org/`.

[2] Ioannis G. Tollis, Giuseppe Di Battista, Peter Eades, and Roberto Tamassia, *Graph drawing*, Prentice Hall Inc., Upper Saddle River, NJ, 1999. Algorithms for the visualization of graphs. MR2064104 (2005i:68067)

[3] Nicolas Bonichon, Mireille Bousquet-Mélou, and Éric Fusy, *Baxter permutations and plane bipolar orientations*, Sém. Lothar. Combin. **61A** (2009/10), Art. B61Ah, 29. MR2734180 (2011m:05023)

[4] John Horton Conway and J. C. Lagarias, *Tiling with Polyominoes and Combinatorial Group Theory*, Journal of Combinatorial Theory **A53** (1990), 183–208.

[5] Richard Kenyon, *An introduction to the dimer model*, School and Conference on Probability Theory, ICTP Lect. Notes, XVII, Abdus Salam Int. Cent. Theoret. Phys., Trieste, 2004, pp. 267–304 (electronic). MR2198850 (2006k:82033)

[6] Tom Leinster, *Higher operads, higher categories*, London Mathematical Society Lecture Note Series, vol. 298, Cambridge University Press, Cambridge, 2004. MR2094071 (2005h:18030)

[7] Saunders Mac Lane, *Categories for the working mathematician*, 2nd ed., Graduate Texts in Mathematics, vol. 5, Springer-Verlag, New York, 1998. MR1712872 (2001j:18001)

[8] William P. Thurston, *Conway's Tiling Groups*, American Mathematical Monthly **97** (October 1990), no. 8, 757–773. Available at `http://www.jstor.org/stable/2324578`.