

Cylinder Diagrams: Visualizing Multi-Parameter Natural Transformations

Niels Voorneveld
Tallinn University of Technology

1 Motivation

Graphical diagrams of composable models for computer programs, for instance handlers of resources, can be used to build an intuition of program equivalence. The idea is that one can move around components in the diagram according to some pre-defined set of intuitive rules, without changing the observed *behaviour* of the program it is representing. Hence, by spotting an equivalence between diagrams, one could intuit an equivalence between the structures it is representing.

A common graphical representation used to this end is that of the *string diagram*, a topological representation of higher-order categorical structures [6]. These can describe morphisms in a monoidal category. For instance, it can illustrate how resources are sent from program to program, where the resource is seen as sent along a string. Informally, the monoidal structure of the category ensures that we can move around the strings without changing the behaviour of the system. If the category moreover is symmetric, we can also cross and braid strings, as long as the source and target of the strings do not change.

Most string diagrams are fundamentally 2-dimensional. One dimension is often reserved for the direction of a process: each morphism has a source and a target, an input and an output which can be connected in various ways. The second dimension can be used for describing simultaneous processes; these are created with the monoidal product and allows us to lay string next to each other, to cross them, and to split them. With 2 dimensions taken up by composition and parallelisation, it does not leave a lot of room for additional structure on top.

One solution to representing more complex structures is to use more dimensions. *Surface diagrams*, as used for instance in [1, 2], can be used to illustrate sheets with string diagrams drawn on top, moving them across each other with the power of depth. There is but one inconvenience; if printed on paper, or projected into a pdf, any 3-dimensional picture is flattened to a 2-dimensional picture. In order to keep all the information of the 3-dimensional model, it is often necessary to use highlights for illustrating things hidden behind other things. This is quite a powerful technique for visualizing definitions, but can become disorienting very quickly in more complex structures. How does one draw three complex layers stacked on top of each other?

In this work, one solution is offered, attempting to use features of both worlds. We use a purely 2-dimensional representation of structures, making sure that every component is either visible or can be inferred from the surroundings. On top of that, we locally add illustrative flourishes to hint towards a third dimension, thereby hopefully building some more intuition for what is being represented. This general idea may not be entirely new. However, this paper will present a specific example of such a method for projecting multi-dimensional models onto a 2-dimensional page such that everything fits neatly next to each other, with no information is lost, whilst still allowing us to imagine the model existing in 3 dimensions.

The particular processes we aim to describe here are natural transformations between endofunctors with multiple parameters. Such a *multi-endofunctor* can be used to modify resources, endowing it with additional structure. As a notable example, notions of computation are often described by a particular type of endofunctor: a *monad* [5]. This takes a resource, and transforms it into a set of computation producing such the resource. When an endofunctor is applied to a resource, we do not have direct access to that resource, as if the endofunctor creates a barrier between this resource and the outside world. Such structures and proofs about them have been beautifully illustrated with 2-dimensional string diagrams in [3]. In this work we aim to complement this representation by allowing an arbitrary number of parallel parameters, e.g. describing parallel programs.

We illustrate endofunctors, which encapsulate resources, using *hollow strings* or *cylinders*. In other words, they are shells which can contain any number of processes, similar to tapes employed in [4]. For example, a functor can be applied to itself, creating a cylinder within a cylinder. Resources can also be put next to each other horizontally, illustrating a monoidal product on multi-endofunctors. A natural transformation between multi-endofunctors is seen as a process between them, and are drawn from top to bottom. We aim to give some intuition on such natural transformations in the way that the cylinders representing the endofunctors are connected.

Importantly, cylinders allow us to switch freely between 2 dimensions and 3 dimensions. In diagrams, cylinders are flattened with all its contents laid out next to each other, hiding no information. When modifying cylinders for the illustration of natural transformations, we can locally use the third dimension. As long as between the operations, we project everything back to the plane, anything temporarily hidden from view can be inferred from the surroundings. For example, when duplicating a resource, we can illustrate this as peeling off a copy of an image (like a sticker), and separate the two copies until they are side by side.

The diagrams in this work are built from a set of pre-drawn tiles. These allow us to repeat specific patterns and keep set distances between components, which should make them easier to interpret. Moreover, tiled diagrams can be edited using tiling software, for example by copy and pasting parts which easily fit together. The images created are then edited in post, using colours to distinguish different of structures.

We will give the basis of how to represent natural transformations between multi-endofunctors. Then we will look at some examples of how they illustrate basic results from the literature, mainly related to monads.

2 Illustrating categories of endofunctors

Consider some symmetric monoidal category \mathcal{C} . We aim to visualize the category \mathcal{E} of multi-endofunctors on \mathcal{C} , which inherits the symmetric monoidal structure of the underlying category. Since any object of \mathcal{C} can be seen as a constant 0-ary endofunctor, and any morphism of \mathcal{C} as a natural transformation between these constant endofunctors, we can use the visualisation of \mathcal{E} to visualise the underlying category \mathcal{C} as well. As a start, let us represent \mathcal{C} first, and add higher-order structures later.

Objects of \mathcal{C} are illustrated with cylinders, which can be coloured to distinguish between them. Morphisms in \mathcal{C} are given by blocks with an input cylinder and an output cylinder. For instance, a morphism f between constant functors A and B is represented as:

$$(f : \frac{A}{B}) = \begin{array}{c} \text{orange cylinder} \\ \text{white block with } f \\ \text{blue cylinder} \end{array}$$

We draw diagrams from top to bottom, in order to ease the readability of domain and codomain expressions (especially for when bracketing is introduced later on). To match this, we write the type of a morphism with fraction notation. The top cylinder represents A , and the bottom cylinder represents B . Note that f separates the two cylinders, which suggests that nothing can pass from the top cylinder to the bottom cylinder.

We can describe the monoidal product of objects in \mathcal{C} by laying objects side-by-side. We look at a concrete morphism; the swap operation for braiding. This takes two arguments and switches their place, and is parametric in its two arguments. Specifically, we represent the swap operation as

$$(S : \frac{X \times Y}{Y \times X}) = \begin{array}{c} \text{orange cylinder} \\ \text{crossing} \\ \text{blue cylinder} \end{array},$$

where we use the suggestive crossing illustration. In this illustration, we see the two cylinders as continuing uninterrupted. Of course the blue cylinder does stop in the 2d projection, but we imagine it going behind the other, and it is only hidden during the operation, not before or after. The fact that the cylinders are uninterrupted suggests that other operations can be dragged across, as illustrated by the following equations:

This is what we mean by *naturality*. A multi-parameter natural transformation is a family of morphisms over a tuple of object parameters. Formally, the swap operation is a natural transformation on the bifunctor \times . Below, we will visualize more general natural transformations, after we introduce general endofunctors. Lastly, since we assume our monoidal structure on \mathcal{C} to be symmetric, we have the usual familiar equations:

Unary endofunctors: A (unary) endofunctor F is drawn using hollow cylinders, which can contain any object of \mathcal{C} within it. The top and the bottom of the cylinder is given a 3-dimensional flourish, but most of the time it will be projected to the plane, showing only the edge of the cylinder on either side. These two sides can be seen as the brackets of the functor application, e.g. as seen in $F(X)$, and have a highlights inside which can be coloured according to functor type. Here we see a functor as applied to objects and morphisms:

$$TA = \text{cylinder with orange core} \quad (f : \frac{A}{B}) = \text{cylinder with orange core and blue ring } f \mapsto (Tf : \frac{TA}{TB}) = \text{cylinder with orange core and blue ring } f$$

Note that F applied to the identity morphism can be illustrated the same way as the left image above, and we cannot distinguish between that and the identity morphism on FA . This internalises the identity property of functors in our graphical representation. Similarly, it is difficult to draw the compositionality property of functors, but one can try by stacking cylinders:

$$(\forall f : \frac{X}{Y}, g : \frac{Y}{Z}. Tg \circ Tf = T(g \circ f) : \frac{TX}{ZY}) = \text{stacked cylinders } f, g = \text{stacked cylinders } g, f \quad RTX = \text{cylinder with orange core and white ring}$$

In the second image above, we see composition of functors, drawn by layering the cylinders and using different colours. The identity of functor composition, the identity functor, is often invisible and not drawn.

Single parameter natural transformations: A morphism in \mathcal{E} between unary endofunctors T to R is a single-parameter natural transformation. A generic morphism can be drawn similar to morphisms in \mathcal{C} , except here we have a parameter cylinder going through the operation without interruption. For example, we can have the following illustration of a natural transformation a , and a morphism of \mathcal{C} passing through:

$$(\forall X. a_X : \frac{TX}{RX}) = \text{cylinder with orange core and blue ring } a \quad \text{With naturality: } (\forall f : \frac{X}{Y}. a_Y \circ Tf = Rf \circ a_X) \quad \text{cylinder with orange core and blue ring } a = \text{cylinder with orange core and blue ring } a$$

Generalisation to multi-endofunctors: Multi-endofunctors can be represented in a lot of ways, by laying multiple cylinders side-by-side, connected to each other in some way (or not connected, if they are separated by a \times). We will not focus on a particular representation here, instead focussing on examples and natural transformations between them. However, we do need to establish how to deal with multiple parameters.

In order to represent natural transformations between multi-endofunctors, we need to be able to connect up the parameters in domain with parameters in the codomain in a variety of ways. To do this, we introduce some extra operations, in the form of natural transformations, which enable us to make full use of this extra dimension. These can be used to represent natural transformations between n -ary endofunctors. We have already seen the swap operation. To this repertoire, we add the copy operation C and the delete operation D :

$$C : \frac{X}{X \times X} = \text{fork} \quad D : \frac{X}{1} = \text{cup}$$

Copying is illustrated by peeling of a one 2-dimensional duplicate and putting the two copies side by side. This is a standardized way of duplicating diagrams without having to deal with the 3-dimensional details of what is being duplicated. Deletion is illustrated by pinching the cylinders to a point. In these notes, we assume that copy and delete are natural, and that they are subject to the unitality and associativity equations:

$$\text{cup} = \text{cup} \quad \text{fork} = \text{fork} \quad \text{fork} = \text{fork} = \text{fork} \quad \text{fork} = \text{fork}$$

These operations can be dualized (put up-side down), creating the comparison co-operation and the generating co-unit. However, these don't exist in the category of sets, and we will leave them out here. When visualising multi-endofunctors, we can glue the cylinders representing the arguments together and mark the glueing with a symbol. A natural transformation between multi-endofunctors can then be illustrated by connecting the parameter cylinders accordingly. For example, we can visualize a bifunctor for coproducts by glueing two cylinders with a plus-symbol, and use the copy operation to represent a distributivity operation:

$$(\delta_{X,Y,Z} : \frac{X \times (Y+Z)}{(X \times Y) + (X \times Z)}) = \text{cylinder with orange core, blue ring, and green ring, with a plus symbol and a diagonal line}$$

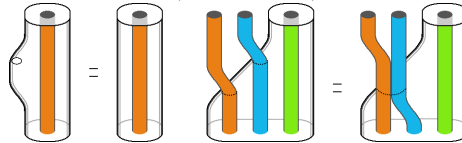
This ends the set-up for the graphical representations using cylinder diagrams. In the rest of these notes, we will look at a variety of example structures which can be represented as natural transformations between multi-endofunctors, to illustrate the versatility of the diagrams. We will be much less verbose, letting the illustrations speak for themselves. To keep it within a theme, we will focus on constructions related to monads.

Example 1: Strong monads

Strength of a functor is a way of pulling an extra resource into the shell of a functor. We illustrate this by piercing a hole in the functor-cylinder and leading through some parameter-cylinder:

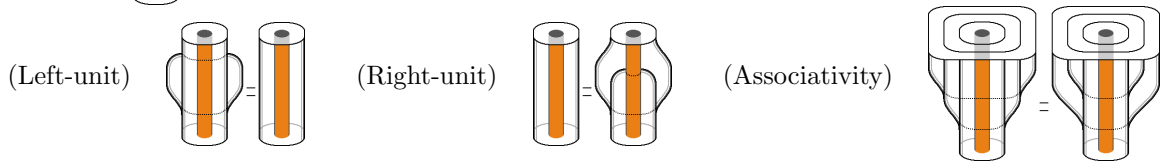
$$\left(\sigma_{X,Y} : \frac{X \times TY}{T(X \times Y)}\right) = \text{diagram}$$

A functor is strong if it has a natural transformation like the one above, which satisfies the following additional equations, telling us that if nothing (meaning 1) goes through the hole we can remove it $\sigma_{1,X} \simeq id_{TX}$, and that we can combine holes $\sigma_{X,Y \times Z} \circ (id_X \times \sigma_{Y,Z}) \simeq \sigma_{X \times Y,Z}$.

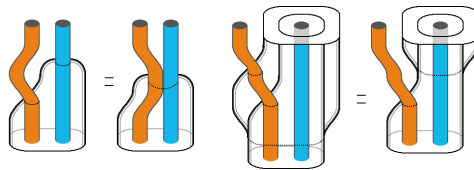



Definition 1. A *monad* is a functor $T = \text{diagram}$ with two natural transformations $(\eta_X : \frac{X}{TX}) = \text{diagram}$ and

$(\mu_X : \frac{TTX}{TX}) = \text{diagram}$, satisfying the following two equations:

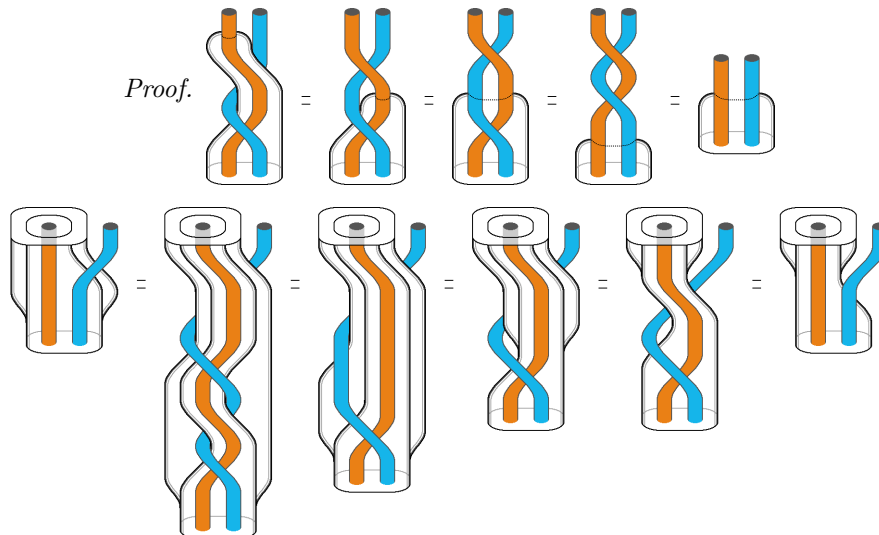


A *strong monad* moreover satisfies the following equations:



Using swap , we define costrength as $(\tau_{X,Y} : \frac{TX \times Y}{T(X \times Y)}) = \text{diagram} = \text{diagram}$.

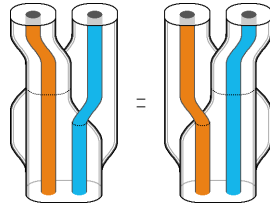
Lemma 1. A *strong monad* is a *costrong monad* (they satisfy the equations for being a strong monad from Definition 1, but replacing strength with costrength).



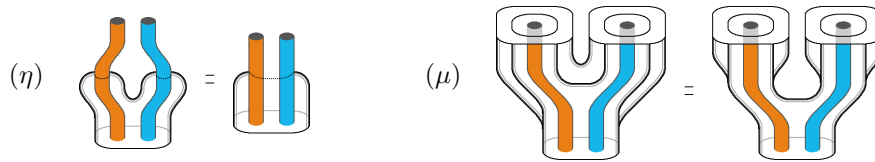
□

Example 2: Commutative monads

Definition 2. A monad is commutative if the following equation holds:

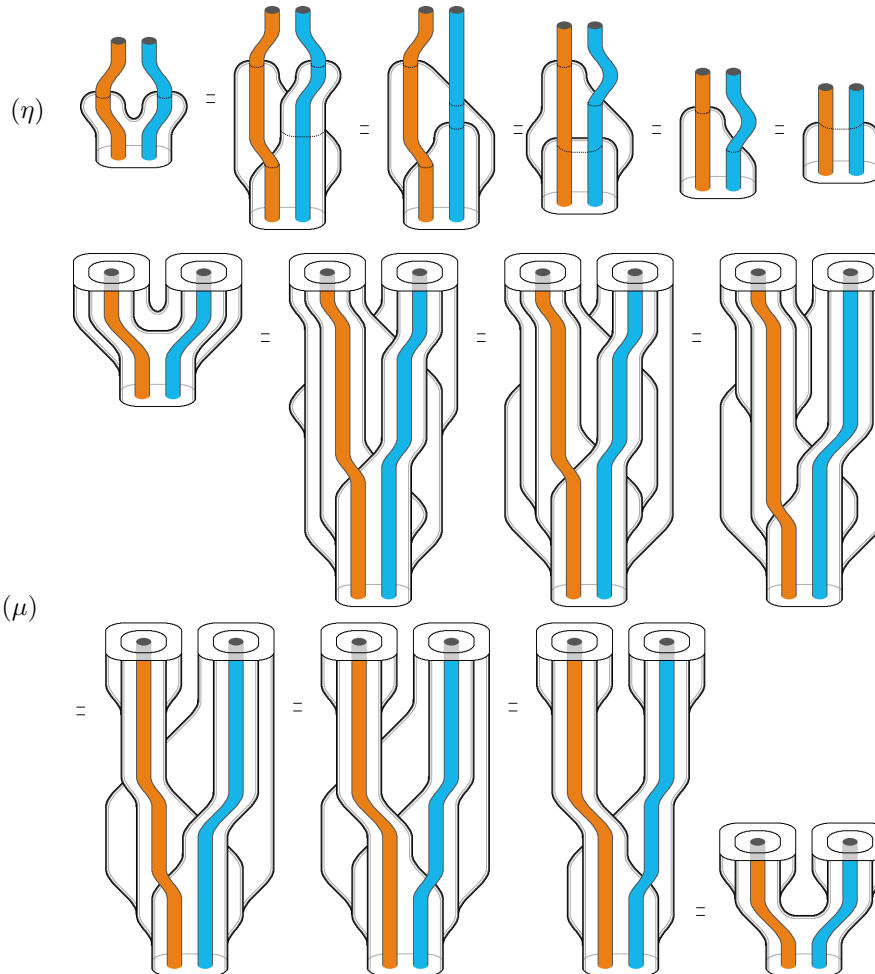


Definition 3. T is a *monoidal monad* if there is a natural transformation $(m : \frac{TX \times TY}{T(X \times Y)}) = \text{Y-shaped diagram}$ satisfying the following two equations:



Lemma 2. Any strong commutative monad is a monoidal monad.

Proof. We define the following natural transformation: $(m : \frac{TX \times TY}{T(X \times Y)}) = \text{Y-shaped diagram} = \text{Y-shaped diagram}$. Using commutativity, we can prove both monoidal monad properties:



□

Example 3: Products of monads

Given two functors T and R , we can take their product

$$(T \times R)(X) = (TX \times RX) = \text{[Diagram of two cylinders side-by-side]}$$

We will illustrate that, if T and R are both monads and copy and delete are natural transformations, then $T \times R$ is a monad. In this case, since we use two different functors in various combinations, we will colour the functor cylinders for ease of understanding. We define the two natural transformations:

- The monad unit $\left(\eta_X^{T \times R} : \frac{X}{TX \times RX}\right) = \text{[Diagram of a single cylinder splitting into two cylinders]}$.

- The monad multiplication $\left(\mu_X^{T \times R} : \frac{T(TX \times RX) \times R(TX \times RX)}{TX \times RX}\right) = \text{[Diagram of two pairs of cylinders merging into one pair]}$.

We show the three equations:

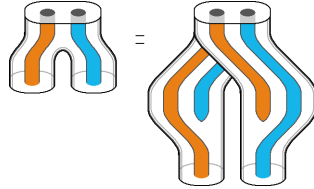
- $\text{[Diagram 1: A pair of cylinders with a monad multiplication structure, showing the unit property. It consists of three stages: a pair of cylinders with a multiplication structure, a pair of cylinders with a unit structure, and finally a pair of cylinders.]}$
- $\text{[Diagram 2: A pair of cylinders with a monad multiplication structure, showing the multiplication property. It consists of three stages: a pair of cylinders with a multiplication structure, a pair of cylinders with a multiplication structure, and finally a pair of cylinders.]}$
- $\text{[Diagram 3: A large diagram showing the associativity of multiplication. It consists of two stages: a pair of cylinders with a multiplication structure, and a pair of cylinders with a multiplication structure.]}$

Example 4: Skew monoidal products from monads

Let T be some endofunctor, we define a binary operation \times on endofunctors as the bifunctor:

$$(X \otimes Y) = TX \times Y = \text{[diagram of two cylinders, one orange and one blue, side-by-side]}$$

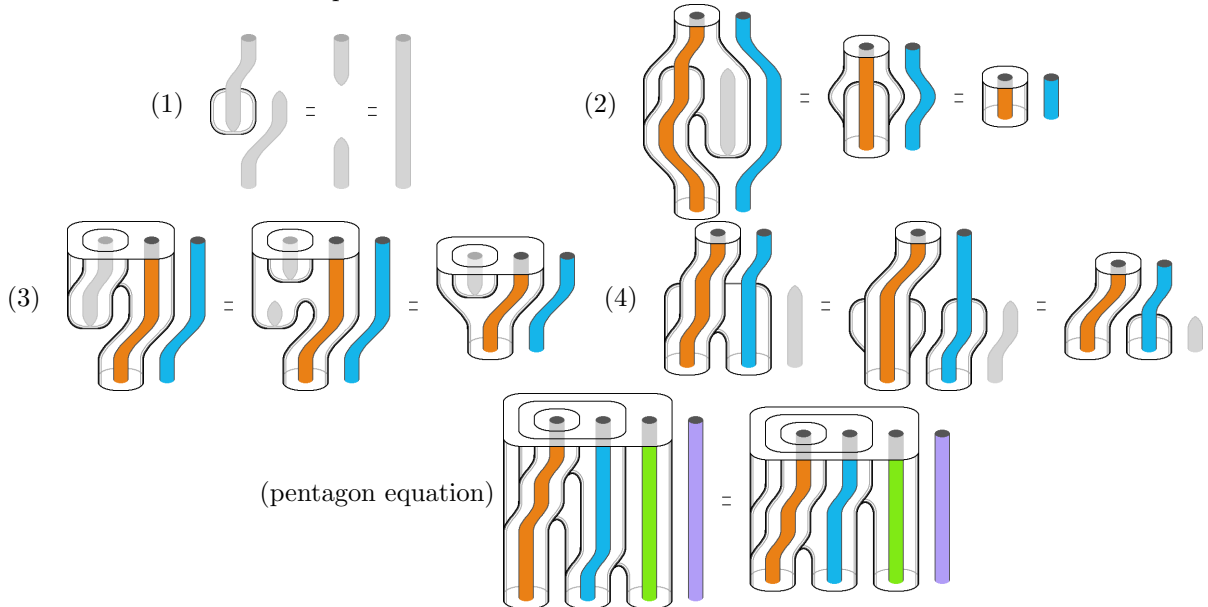
It holds that this bifunctor forms a skew monoidal product in the sense of Szlachányi 2012 [7] if and only if the functor T is a monad. We show this in one direction: supposing T is a monad, we create a skew monoidal structure on \otimes . Using copy and delete, we can define comonoidal co-operation on T as:



As unit, we use the constant endofunctor on the unit $\mathbf{1}$ of \mathcal{C} , which is itself the unit of the product on endofunctors. Normally, we would remove such units from our picture. But to clarify the connection with skew monoidal categories, we include them in the formulation of the skew monoidal structure as light grey cylinders. New units can be introduced and glued to existing units freely.

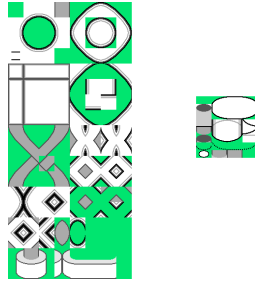
- Let skewed associativity be $\left(\alpha : \frac{(X \otimes Y) \otimes Z}{X \otimes (Y \otimes Z)}\right) = \text{[diagram of three cylinders, one orange, one blue, one green, with a grey unit cylinder]} ,$ using the co-operation and multiplication.
- Let skewed left-unitality be $\left(\lambda : \frac{\mathbf{1} \otimes X}{X}\right) = \text{[diagram of a grey unit cylinder and an orange cylinder]} .$
- Let skewed right-unitality be $\left(\rho : \frac{X}{X \otimes \mathbf{1}}\right) = \text{[diagram of an orange cylinder and a grey unit cylinder]} .$

We show the five coherence equations.



3 Final thoughts

The images in this work have been generated using image tiling software, using images from the following two tilesets, respectively containing 8 by 18 tiles with 32x32 pixels, and 4 by 8 tiles with 32x16 pixels:



Most pictures are built in three layers onto a white background, where the green in the tilesets marks transparent areas in the tiles. First two layers use the first tileset, consisting of a base layer and an extra layer for overlaying other components as for instance done in swapping and copying. The third layer is used for additional 3-dimensional flourish, mostly at the top and bottom of the diagrams. After tiling, the images undergo some post-editing, which mostly involves the colouring of components and some minimal rescaling of components.

This work was inspired by the need to visualise dependencies of parallel processes. Though it did not fit within the scope of this short paper, the diagrams can be used to represent natural transformations describing *concurrent interleaving processes* and *program-environment interaction laws*. Categorical properties of such operation tend to be quite abstract and require some effort to interpret. Hopefully, the method for visualisations presented here will be a helpful aid for building intuition about these concepts, representing unevaluated computations as barriers which need to be scheduled or handled.

Similar ways of representing similar concepts have appeared throughout the literature. However, this specific way of using cylinders which canonically shift between 2D and 3D, and in particular using tiles to build them, is believed to be novel. This final way of representing natural transformations was influenced by discussions with other researchers. Thanks to Ed Morehouse for his explanation and illustrations of 3-dimensional surface diagrams for representing natural transformations, and Cole Comfort for some of his applications surface diagrams. Thanks to Pawel Sobocinski, who had been working on a similar, yet unpublished, visualisation style called ribbon semantics. And thanks to others for helpful suggestions and feedback, including but not limited too; Mario Roman, Elena Di Lavore, Chad Nester and Philipp Joram.

References

- [1] John Barrett, Catherine Meusburger, and Gregor Schaumann. Gray categories with duals and their diagrams. 11 2012.
- [2] Lawrence Dunn and Jamie Vicary. Surface proofs for nonsymmetric linear logic (extended abstract). *Electronic Proceedings in Theoretical Computer Science*, 238:33–43, Jan 2017.
- [3] Daniel Marsden. Category theory using string diagrams, 2014.
- [4] Micah Blake McCurdy. Graphical methods for tannaka duality of weak bialgebras and weak hopf algebras in arbitrary braided monoidal categories. *Theory and Applications of Categories*, 26:233–280, 2012.
- [5] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, July 1991.
- [6] Ross Street. Low-dimensional topology and higher-order categories. In *In Proceedings of CT95*, 1995.
- [7] Kornél Szlachányi. Skew-monoidal categories and bialgebroids. *Advances in Mathematics*, 231(3-4):1694–1730, Oct 2012.