

Resourceful traces for commuting processes

Matthew Earnshaw ✉

Tallinn University of Technology, Estonia

Chad Nester ✉

University of Tartu, Estonia

Mario Román ✉

University of Oxford, United Kingdom

Abstract

We show that, when the actions of a Mazurkiewicz trace are considered not merely as atomic (i.e., mere names) but transformations from a specified type of inputs to a specified type of outputs, we obtain a novel notion of presentation for effectful categories (also known as generalised Freyd categories), a well-known algebraic structure in the semantics of side-effecting computation. Like the usual representation of traces as graphs, our notion of presentation gives rise to a graphical calculus for effectful categories. We use our presentations to give a construction of the commuting tensor product of free effectful categories, capturing the combination of systems in which the actions of each must commute with one another, while still permitting exchange of resources.

2012 ACM Subject Classification Theory of computation → Concurrency; Theory of computation → Categorical semantics

Keywords and phrases Mazurkiewicz traces, premonoidal categories, monoidal categories, effectful categories

Funding *Matthew Earnshaw*: Estonian Research Council grant PRG1210.

Chad Nester: Estonian Research Council grant PRG2764.

Mario Román: Air Force Office of Scientific Research (AFOSR) award number FA9550-21-1-0038; Advanced Research + Invention Agency (ARIA), Safeguarded AI Programme.

1 Introduction

Mazurkiewicz traces [5, 16] provide a simple but powerful model of concurrent systems. Traces are a generalization of words, in which specified pairs of symbols (thought of as actions) can commute. Commuting actions a and b are *independent*: their possible concurrent execution ab is observationally indistinguishable from ba .

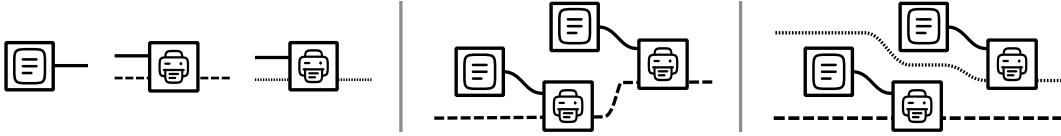
Just as free monoids are the algebraic structure formed by words, *free partially commutative monoids* [8] or *trace monoids* [16, 24] are the algebraic structure formed by (Mazurkiewicz) traces. Traces therefore permit an algebraic approach to the analysis of concurrent systems, by analogy with the use of algebraic methods in automata theory.

Mazurkiewicz states that intuitively, “*actions* are state transformations of some resources of a system” and that “actions are independent if they act on disjoint sets of resources” [15]. We contend that this view conflates two notions of resource that might be present in a concurrent system, and it is the remit of this paper to show that by teasing them apart, we obtain a richer algebraic structure.

On the one hand are shared resources corresponding to definite noun phrases, such as “*the database*”, “*the memory location*”, or “*the printer*”. These indeed give rise to relations of (in)dependence between actions. In this paper, we shall refer to such resources as *devices*. On the other hand are the kinds of resources and their transformations axiomatized by *monoidal categories*, as in Coecke, Fritz and Spekkens [4]. These are resources which act like *types*, and generally correspond to indefinite noun phrases such as “*a database query*”, “*an integer*”,

or “a message”. Given any two such resources, we can always consider their conjunction “in parallel”, and they do not necessarily lead to relations of (in)dependence between actions.

For example, consider Figure 1, which introduces a running example, in our graphical notation. The left-hand side of the figure corresponds to our analogue of the alphabet of actions underlying a trace: we call this an *effectful graph*. In our example, the effectful graph contains a process corresponding to the action of *emitting a document*, with a single output string, and two processes corresponding to the action of *printing* received data on two distinct printers. Note that these actions are no longer *atomic*: they have strings corresponding to resources (depicted as solid strings, in general annotated with particular resources), that here indicate the data to be printed. Crucially, the two printing actions are annotated by distinct *devices*, depicted as patterned strings. Intuitively, a *resourceful trace* is a *string diagram* (directed from a left to a right boundary) built by plugging generators together, while only allowing each device string to appear at most once in each vertical section.



■ **Figure 1** (Left) An effectful graph. (Center) Resourceful trace over the first two elements of the effectful graph, i.e. involving only one printer. (Right) Resourceful trace involving two printers: printings are now independent of one another.

If we only admit *one* printer in our system, printing actions must occur in a specific order, and this is captured by the topology of the trace, as in the center of Figure 1. However, if we admit both printers, printing can occur concurrently, and this is again captured by the topology of the traces, such as on the right of Figure 1, where the printing actions may slide past one another or *interchange*. We shall see in Section 5 that this example in fact arises by a canonical combination of the “theory of a printer” with itself, called the *commuting tensor product*, generalizing the commuting tensor product of algebraic theories [9].

The collection of all resourceful traces over an effectful graph assembles into an *effectful category* [22], also known as generalized Freyd categories [10, 20], a well-known algebraic structure in the semantics of effectful computation. Indeed, one of the results of this paper is that resourceful traces are the morphisms of free effectful categories.

Related work

That Mazurkiewicz traces can be seen as morphisms in certain free monoidal categories was elaborated by Sobociński and the first author [7]. By moving to effectful categories, we sharpen this viewpoint, recovering Mazurkiewicz traces precisely as the morphisms of free effectful categories with no resources. The construction of free effectful categories over effectful graphs with a single device was given by the third author, following Jeffrey [12, 21, 22]. The construction of the commuting tensor product of effectful categories appeared in [6], in terms of string diagrams, covering the case of morphisms supported by a finite number of devices. The central results of this paper are new, namely the existence of an adjunction between effectful graphs and effectful categories (Corollary 42), and Theorem 47 on the commuting tensor product of free effectful categories.

Outline

Section 2 recalls the notion of Mazurkiewicz trace, and their graphical representation. Section 3 introduces *effectful graphs* and *effectful categories*, via their underlying *device graphs* and *monoidal graphs*, generalizing the notion of distribution of an alphabet of actions. We give a construction of the free effectful category over an effectful graph, whose morphisms are *resourceful traces*. In Section 4, we show how the cliques construction from the theory of traces generalizes to effectful categories, and exhibits the free construction as a left adjoint. Finally, Section 5 gives a construction of the commuting tensor product of free effectful categories, capturing the combination of systems in which the morphisms of each are forced to commute with one another, even while they may exchange resources.

2 Mazurkiewicz traces

In this section, we recall the basic definitions of trace theory, including the graphical presentation of traces. For more details, we refer to Mazurkiewicz, Hoogetboom and Rozenberg [5]. For this section, we fix a set of actions Σ .

► **Definition 1.** A dependency relation, $D \subseteq \Sigma \times \Sigma$, is a reflexive, symmetric relation. Dependency relations form a preorder, Dep_Σ , with order the inclusion of relations.

Intuitively, a dependency relation specifies actions which should not occur concurrently, because they have some kind of dependency on each other, or the potential to *interfere*, such as writing to the same location in memory.

► **Definition 2.** Given a dependency relation $D \subseteq \Sigma \times \Sigma$, let $\equiv_D$ be the least congruence on Σ^* such that for every pair of actions $a, b \in \Sigma$, $(a, b) \notin D$ implies $ab \equiv_D ba$. The free partially commutative monoid or *trace monoid* generated by D is the quotient monoid $\Sigma^* / \equiv_D$. An element of the trace monoid is a Mazurkiewicz trace (or simply trace) over (Σ, D) .

A trace is thus an equivalence class of words up to commutation of *independent* actions. Another construction of traces starts from *distributions* of the set of actions:

► **Definition 3.** A distribution of Σ is a function $\text{dev} : \Sigma \rightarrow \mathcal{P}(\{1, \dots, k\})$ for some $k \geq 1$. Distributions form a preorder Dist_Σ with $\text{dev} \geq \text{dev}'$ if and only if for all $a, b \in \Sigma$,

$$\text{dev}'(a) \cap \text{dev}'(b) \neq \emptyset \implies \text{dev}(a) \cap \text{dev}(b) \neq \emptyset.$$

Classically, $\text{dev}(\sigma)$ is known as the set of *locations* of σ . In line with the terminology introduced in the following, we call $\text{dev}(\sigma)$ the set of *devices* of σ . In terms of concurrency, we might consider $\text{dev}(\sigma)$ to be the set of shared resources on which σ depends, such as *memory locations*, *execution threads*, *runtimes* or *peripherals*.

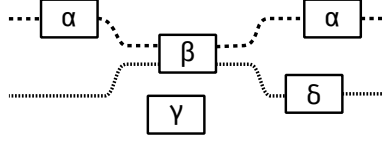
We can represent a distribution graphically by introducing nodes (depicted as boxes) corresponding to actions in Σ , with interfaces (depicted as patterned strings) corresponding to the devices assigned to the action, as for example in Figure 2.



■ **Figure 2** Graphical representation of a distribution: α and β share one device, and β and γ share one device, whereas γ has no devices.

Traces are the diagrams that we can build by plugging these components together, subject to the restriction that each device string occurs exactly once in each vertical slice of the

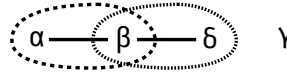
diagram, such as in Figure 3. This is formalized by the algebra of (pre)monoidal categories, as we shall see in the next section. The collection of all such diagrams also forms a monoid, isomorphic to the **trace monoid**. More details can be found in the work of Sobociński and the first author [7], which builds on earlier work on the representation of traces as graphs [5].



■ **Figure 3** Graphical representation of a trace over the distribution in Figure 2. As an equivalence class of words in the **trace monoid**, this trace would include for example $\gamma\alpha\beta\alpha\delta$ and $\alpha\beta\gamma\delta\alpha$: independence is captured by *sliding* actions, while keeping the boundaries fixed.

A classical construction lets us move between **dependency relations** and **distributions**: devices correspond to (non-trivial) *maximal cliques* in the graph of the **dependency relation**. For example, the dependency relation depicted in Figure 4 gives rise to the distribution in Figure 2. In Section 4, we shall apply a similar construction to *effectful categories*.

► **Definition 4.** *The graph of a **dependency relation** $\Sigma \subseteq D \times D$ has vertices the elements of Σ and an edge (a, b) for every $(a, b) \in D$.*



■ **Figure 4** Graph of a dependency relation, whose indicated non-trivial maximal cliques give rise to the distribution of Figure 2.

► **Proposition 5** (§4.1, [7]). *Every **dependency relation** D on Σ gives rise to a **distribution** of Σ by taking a bijection of $\{1, \dots, k\}$ with the non-trivial maximal cliques of the graph of D , with dev sending an action to the subset of such cliques to which it belongs. This extends to a **Galois insertion**, **cliques**: $\text{Dep}_\Sigma \hookrightarrow \text{Dist}_\Sigma$.*

Finally, we remark that any monoid has an underlying **distribution**, which can be found using the maximal cliques construction. A monoid M has **dependency relation** D on its elements defined by $(a, b) \in D$ just when a and b do not commute in M , and taking non-trivial maximal cliques in the graph of this **dependency relation** defines a **distribution** of M . We shall extend this idea to *effectful categories* in Section 4.

3 Resourceful traces and free effectful categories

In traces, actions are merely atomic *names*, equipped with a set of devices. In this section, we enrich the language of traces by allowing actions to additionally have input and output types, which we think of as resources that may be shared between actions. Actions therefore become *morphisms*, with a designated source and target.

In Section 3.1, we introduce the resourceful analogue of the **distribution** of a set of actions, which we call an **effectful graph**. In an **effectful graph**, in addition to a set of devices, every action is equipped with input and output types of resources. Then, in Section 3.3, we show that just as a **distribution** generates a free partially commutative monoid, effectful graphs generate free *effectful categories* [22], also known as *generalized Freyd categories* [10, 20].

3.1 Effectful graphs

Effectful graphs refine distributions: not only by equipping actions with input and output types, but additionally by designating some *central* or *pure* actions, with only the impure actions being potentially able to interfere. In programming language theory, this is the division between *values* and *computations* [18, 14].

One reason for this can be seen already at the level of trace monoids. Morphisms of trace monoids are, a priori, simply morphisms of monoids, but these need not preserve centrality of actions. That is, if an action α commutes with all actions in a trace monoid M (i.e., α is *central* in M), then its image under a monoid homomorphism $M \rightarrow N$ need not commute with all actions of N . Thus, certain actions that we might consider to be *pure* might change their role under a translation of systems. However, restricting the notion of morphism to require preservation of centrality is too strong, since it is possible for computations to be incidentally central: an example may be found in Staton and Levy [23, §5.2].

To overcome this, we explicitly designate which actions we want to consider as pure, and ask that centrality of just these actions is preserved. The chosen set of *pure* actions is specified by a *monoidal graph*:

► **Definition 6.** A monoidal graph M consists of a set of objects, V_M ; a set of arrows, E_M ; and a pair of functions $\delta_0, \delta_1 : E_M \rightarrow V_M^*$ assigning source and target lists of objects to each arrow. We can equivalently present a monoidal graph by a set of objects V_M , and sets $M(X_1, \dots, X_n; Y_1, \dots, Y_m)$ of arrows for each pair of lists of objects: we shall find both perspectives useful.

We can picture a monoidal graph as on the left of Figure 5, where boxes depict the arrows, and their associated strings designate the source and targets. We use *solid* strings for the objects of a monoidal graph.

► **Definition 7.** A morphism of monoidal graphs $\alpha : M \rightarrow N$ comprises two functions $\alpha_E : E_M \rightarrow E_N$ and $\alpha_V : V_M \rightarrow V_N$ commuting with δ_0 and δ_1 , that is, $\alpha_V^* \circ \delta_i = \delta_i \circ \alpha_E$.

Equivalently, a morphism is specified by a function α on objects, and functions

$$M(X_1, \dots, X_n; Y_1, \dots, Y_m) \rightarrow N(\alpha X_1, \dots, \alpha X_n; \alpha Y_1, \dots, \alpha Y_m).$$

It is easy to check that morphisms of monoidal graphs compose, the composite given by composing their underlying actions on arrows and objects, and that we have identities. Denote by MonGraph the category of monoidal graphs and their morphisms.

The set of morphisms that may be equipped with devices, specifying dependencies between morphisms, is given by a *device graph*, which is simply a monoidal graph together with extra data associating a set of devices to each morphism:

► **Definition 8.** A device graph C comprises

- a monoidal graph $|C|$,
- a set $\mathcal{D}_C$ of devices, and
- a function $\text{dev}_C : E_{|C|} \rightarrow \mathcal{P}(\mathcal{D}_C)$ assigning each arrow in $|C|$ to a subset of the devices. We shall simply write dev when the device graph C is clear from context.

We can depict a device graph as on the right of Figure 5. Patterned strings denote the devices, which need not occur in a particular order. In contrast to resource strings, device strings must *thread through* the process: this enforces an order between dependent processes.



■ **Figure 5** Left: Monoidal graph with arrows $\alpha : X \rightarrow \varepsilon$, $\beta : XY \rightarrow X$. Right: Device graph with the same underlying monoidal graph as on the left. α and β share a device, but β has an additional device. The devices of an action all appear in both the source and target, but need not occur in a specific order.

► **Example 9.** A device graph in which the underlying monoidal graph has an empty set of objects and set of arrows Σ is precisely a distribution of Σ in the sense of Definition 3, although we allow the set of devices to be infinite. In practice, a finite set of devices often suffices. For example, if the device graph has a finite number of arrows, then an infinite number of devices can be shown to be redundant.

► **Definition 10.** Morphisms f and g in a device graph C are said to be orthogonal, denoted $f \perp_C g$ just when they share no devices, $\text{dev}(f) \cap \text{dev}(g) = \emptyset$. Note that any f with no devices is orthogonal to every g . We write $f \perp g$ when C is clear from context, and $f \not\perp g$ when f and g are not orthogonal.

A morphism of device graphs must preserve orthogonality,

► **Definition 11.** A morphism of device graphs, $\alpha : C \rightarrow D$, comprises

- a morphism of underlying monoidal graphs, $\alpha : |C| \rightarrow |D|$,
- such that for all arrows $f, g \in C$, if $f \perp_C g$ then $\alpha(f) \perp_D \alpha(g)$.

► **Proposition 12.** Device graphs and their morphisms form a category DevGraph .

Proof. Composition of morphisms is given by composition of their underlying morphisms of monoidal graphs, and identities by the identities on monoidal graphs. It remains to check that the orthogonality condition is satisfied by a composite $\delta \circ \beta : \mathcal{G} \rightarrow \mathcal{H} \rightarrow \mathcal{K}$. Indeed, if $f \perp_{\mathcal{G}} g \Rightarrow \beta(f) \perp_{\mathcal{H}} \beta(g)$ for all $f, g \in \mathcal{G}$ and $h \perp_{\mathcal{H}} i \Rightarrow \delta(h) \perp_{\mathcal{K}} \delta(i)$ for all $h, i \in \mathcal{H}$ then we have

$$f \perp_{\mathcal{G}} g \Rightarrow \beta(f) \perp_{\mathcal{H}} \beta(g) \Rightarrow \delta(\beta(f)) \perp_{\mathcal{K}} \delta(\beta(g)) = (\delta \circ \beta)(f) \perp_{\mathcal{K}} (\delta \circ \beta)(g)$$

for all $f, g \in \mathcal{G}$ as required. ◀

We now have the ingredients required to define *effectful graphs*,

► **Definition 13.** An effectful graph $\mathcal{G} : V \rightarrow C$ comprises

- a monoidal graph V ,
- a device graph C over the same set of objects as V ,
- an identity-on-objects morphism of monoidal graphs $\mathcal{G} : V \rightarrow |C|$,
- such that $\text{dev}(\mathcal{G}v) = \emptyset$ for all arrows v in V .

We usually think of V in terms of its image in C , and thus when we speak of a *morphism in an effectful graph*, we are referring to the arrows of the device graph C .

► **Example 14.** In the effectful graph in Figure 1, the monoidal graph comprises solely the “document” generator, while the device graph includes all three generators, with the morphism of monoidal graphs simply including the “document” generator.

► **Definition 15.** A morphism of effectful graphs $(\alpha_0, \alpha) : (\mathcal{G} : V \rightarrow C) \rightarrow (\mathcal{H} : W \rightarrow D)$ comprises

- a morphism of monoidal graphs $\alpha_0 : V \rightarrow W$, and
- a morphism of device graphs $\alpha : C \rightarrow D$,
- such that the square of morphisms of monoidal graphs commutes, $|\alpha| \circ \mathcal{G} = \mathcal{H} \circ \alpha_0$.

Since both MonGraph and DevGraph are categories, it follows easily that:

► **Proposition 16.** *Effectful graphs and their morphisms form a category EffGraph.*

Note that effectful graphs generalize the *effectful polygraphs* of Sobociński and the third author [22], precisely by allowing each action to have a different *set* of devices, rather than all actions having a single, global device. The construction of effectful categories in the next section also refines that from *effectful polygraphs* [22], in that it builds-in extra independence equations between processes.

3.2 Effectful categories

Effectful graphs generate *effectful categories*. An effectful category comprises a monoidal category $\mathbb{V}$, a premonoidal category $\mathbb{C}$, and an identity-on-objects premonoidal functor $\mathbb{V} \rightarrow \mathbb{C}$. In this paper, we work with *strict* versions of these notions, since not only are the free such structures *strict*, but coherence theorems exist which state that any non-strict (pre)monoidal category is equivalent to a strict one [19].

Just as with effectful graphs, the idea is that $\mathbb{V}$ is a category of pure actions (generated by a monoidal graph), $\mathbb{C}$ is a category of effectful actions (generated by a device graph), and the functor is thought of as an inclusion of $\mathbb{V}$ into $\mathbb{C}$. Let us build up to their definition.

► **Definition 17.** *A strict monoidal category is a category $\mathbb{V}$ equipped with*

- a functor $\otimes : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{V}$, called the *monoidal product*,
- a distinguished object I , called the *monoidal unit*,
- such that $A \otimes (B \otimes C) = (A \otimes B) \otimes C$ and $A \otimes I = A = I \otimes A$,
- and such that the following equations hold for all morphisms f, g, h :

$$f \otimes (g \otimes h) = (f \otimes g) \otimes h \qquad 1_I \otimes f = f = f \otimes 1_I$$

Premonoidal categories refine monoidal categories by only requiring the monoidal product to be functorial separately in each variable: in particular, there is no operation combining morphisms in parallel.

► **Definition 18.** *A strict premonoidal category is a category, $\mathbb{C}$, equipped with functors $(A \ltimes -) : \mathbb{C} \rightarrow \mathbb{C}$ (the “left-whiskering” by A) and $(- \rtimes A) : \mathbb{C} \rightarrow \mathbb{C}$ (the “right-whiskering” by A), for every object A of $\mathbb{C}$, and a distinguished object I (the “monoidal unit”), such that,*

- $A \ltimes B = A \rtimes B$, allowing us to denote this object as $A \otimes B$,
- $A \otimes (B \otimes C) = (A \otimes B) \otimes C$ and $A \otimes I = A = I \otimes A$,
- and such that the following equations (Figure 6) hold, stating coherence of left- and right-whiskering with each other and with the monoid structure,

$$\begin{aligned} (A \ltimes f) \rtimes B &= A \ltimes (f \rtimes B) & I \ltimes f &= f = f \rtimes I \\ (A \otimes B) \ltimes f &= A \ltimes (B \ltimes f) & f \rtimes (A \otimes B) &= (f \rtimes A) \rtimes B \end{aligned}$$

■ **Figure 6** Equations for left and right whiskerings in a strict premonoidal category.

Note that every monoidal category is a premonoidal category in which partial applications of the monoidal product define the left- and right- whiskerings. The key equations that hold for every pair of morphisms in a monoidal category, but not generally in a given premonoidal category, are those expressing *interchange* (i.e., *commutation*) of morphisms:

- **Definition 19.** For morphisms $f : A \rightarrow B$ and $g : A' \rightarrow B'$ in a premonoidal category $\mathbb{C}$,
 - we write $f \parallel g$ in case $(f \rtimes A')(B \ltimes g) = (A \ltimes g)(f \rtimes B')$,
 - we say that f and g *interchange* in case $f \parallel g$ and $g \parallel f$,
 - we say that f and g *interfere*, written $f \nparallel g$, in case they do not interchange,
 - we say that f is *central* in case it *interchanges* with every morphism of $\mathbb{C}$.

Graphically, interchange of morphisms is precisely the *sliding* of morphisms past one another, as discussed for example in Figure 3.

- **Definition 20.** Let $\mathbb{C}$ and $\mathbb{D}$ be strict premonoidal categories. A strict premonoidal functor is a functor $F : \mathbb{C} \rightarrow \mathbb{D}$ that

- maps objects as a monoid homomorphism, $F(A \otimes B) = F(A) \otimes F(B)$ and $F(I) = I$,
 - and preserves whiskerings, $F(A \ltimes f) = F(A) \ltimes F(f)$ and $F(f \rtimes A) = F(f) \rtimes F(A)$.
- When $\mathbb{C}$ and $\mathbb{D}$ are moreover strict monoidal categories, we call this a strict monoidal functor.

Finally, we reach the definition of an *effectful category*:

- **Definition 21.** A (strict) effectful category $\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}$ comprises

- a strict monoidal category $\mathbb{V}$,
- a strict premonoidal category $\mathbb{C}$ with $\mathbb{C}_{\text{obj}} = \mathbb{V}_{\text{obj}}$, and
- an identity-on-objects strict premonoidal functor $\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}$,
- such that the image of $\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}$ is central.

- **Definition 22.** A morphism of (strict) effectful categories, $(\alpha_0, \alpha) : (\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}) \rightarrow (\mathcal{F} : \mathbb{W} \rightarrow \mathbb{D})$, comprises a strict monoidal functor $\alpha_0 : \mathbb{V} \rightarrow \mathbb{W}$ and a strict premonoidal functor $\alpha : \mathbb{C} \rightarrow \mathbb{D}$ such that $\alpha \circ \mathcal{E} = \mathcal{F} \circ \alpha_0$ in PremonCat (eliding the inclusion of $\text{MonCat} \hookrightarrow \text{PremonCat}$).

From the fact that MonCat and DevGraph are categories, it follows easily that:

- **Proposition 23.** *Effectful categories and their morphisms form a category EffCat .*

3.3 Free effectful category over an effectful graph

Our goal now is to define a functor $\mathcal{F} : \text{EffGraph} \rightarrow \text{EffCat}$. We will see in the next section that this functor has a right adjoint, and so truly constructs the *free* effectful category on an effectful graph. We begin by constructing the free premonoidal category over a device graph.

- **Definition 24.** For a device graph G , we construct a premonoidal category $\mathcal{F}G$ on G as follows. The underlying category has,

- set of objects V_G^* , the free monoid on V_G , whose elements are lists. We shall denote by $\otimes$ the concatenation of lists.
- set of morphisms $E_{\mathcal{F}G}/\equiv$, where $E_{\mathcal{F}G}$ is the set inductively defined by rules for identities, the whiskering of generating arrows, and composition,

$$\begin{array}{c}
 \text{ID} \\
 \hline
 X \in V_G^* \\
 \hline
 1_X : X \rightarrow X
 \end{array}
 \quad
 \begin{array}{c}
 \text{WHISK} \\
 \hline
 X, Y \in V_G^* \quad g : W \rightarrow Z \in E_G \\
 \hline
 X \triangleright g \triangleleft Y : X \otimes W \otimes Y \rightarrow X \otimes Z \otimes Y
 \end{array}
 \quad
 \begin{array}{c}
 \text{COMP} \\
 \hline
 u : P \rightarrow Q \quad v : Q \rightarrow R \\
 \hline
 u \circ v : P \rightarrow R
 \end{array}$$

■ and $\equiv$ is the least congruence for $\circledcirc$ generated by,

$$(u \circledcirc v) \circledcirc w = u \circledcirc (v \circledcirc w) \qquad 1_X \circledcirc u = u = u \circledcirc 1_Y$$

whenever these are well typed, along with equations allowing (whiskerings of) orthogonal generating arrows to interchange, i.e. for every pair, $f: U \rightarrow V$ and $g: U' \rightarrow V'$, of arrows in E_G that are orthogonal, $\text{dev}(f) \cap \text{dev}(g) = \emptyset$, and each triple of objects X, Y, Z in V_G^* ,

$$(X \triangleright f \triangleleft Y \otimes U \otimes Z) \circledcirc (X \otimes V \otimes Y \triangleright g \triangleleft Z) = \\ (X \otimes U' \otimes Y \triangleright g \triangleleft Z) \circledcirc (X \triangleright f \triangleleft Y \otimes V' \otimes Z).$$

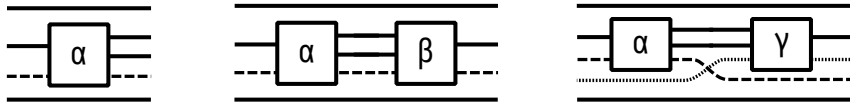
It is immediate that this data forms a category. For the *premonoidal* structure, we define I to be the empty list, left and right whiskerings act on objects by list concatenation (denoted $\otimes$), and their action on morphisms is defined inductively by

$$(X \ltimes f) := \begin{cases} 1_{X \otimes Y} & \text{if } f = 1_Y, \\ (X \ltimes g) \circledcirc (X \ltimes h) & \text{if } f = g \circledcirc h, \\ X \otimes Y \triangleright f' \triangleleft Z & \text{if } f = Y \triangleright f' \triangleleft Z, \end{cases}$$

$$(f \rtimes X) := \begin{cases} 1_{Y \otimes X} & \text{if } f = 1_Y, \\ (g \rtimes X) \circledcirc (h \rtimes X) & \text{if } f = g \circledcirc h, \\ Y \triangleright f' \triangleleft Z \otimes X & \text{if } f = Y \triangleright f' \triangleleft Z. \end{cases}$$

It is easily verified that these mappings are well-defined, functorial, and satisfy the required coherence equations of Definition 18.

The morphisms of $\mathcal{FG}$ are *resourceful traces* over G , and can be depicted intuitively as “string diagrams” (from an input boundary to an output boundary), built by combining whiskered generators of G via the operations of premonoidal categories. Generators correspond to boxes as in Figure 5, identity morphisms correspond to strings, composition is given by joining resource strings and any matching device strings, leading to a morphism with the union of the two sets of devices, and whiskering by juxtaposing strings with boxes. Whiskering by the empty list is simply depicted by drawing no whiskering string. We illustrate this in Figure 7. The requirement that every device string appear at most once in every vertical slice through the diagram, as in Section 2, is enforced by the fact that we have no operation for combining morphisms in parallel. The following lemma shows that interchange of generating arrows extends to the morphisms of $\mathcal{FG}$ via the device assignment described above.



■ **Figure 7** (Left to right): Whiskering of a generating arrow by a resource, composition of morphisms with a matching device, composition of morphisms with distinct devices. In each composition, we obtain a new morphism whose devices are the union of the components. Types of the resource strings are omitted here for clarity.

► **Lemma 25.** Consider the inductive extension of the device assignment of a device graph G to morphisms of $\mathcal{FG}$ (Definition 24),

$$\text{dev}_{\mathcal{FG}}(f) := \begin{cases} \emptyset & \text{if } f = 1_A, \\ \text{dev}_G(f') & \text{if } f = X \triangleright f' \triangleleft Y, \\ \text{dev}_{\mathcal{FG}}(g) \cup \text{dev}_{\mathcal{FG}}(h) & \text{if } f = g \circledcirc h. \end{cases}$$

Let f, g be two morphisms of $\mathcal{FG}$. If f and g are orthogonal, $\text{dev}_{\mathcal{FG}}(f) \cap \text{dev}_{\mathcal{FG}}(g) = \emptyset$, then $f \parallel g$ and $g \parallel f$.

Proof. See Section A. ◀

► **Example 26.** As noted in Example 9, a device graph G with empty set of objects is precisely a **distribution** of an alphabet. In this case, $\mathcal{FG}$ has a single object, the empty list, and hence is a monoid (with trivial whiskerings). This monoid is isomorphic to the trace monoid over the corresponding distribution.

► **Lemma 27.** Each morphism of device graphs, $\alpha : G \rightarrow H$, induces a strict premonoidal functor, $\mathcal{F}(\alpha) : \mathcal{FG} \rightarrow \mathcal{FH}$, defined as follows. The action on objects is the lifting of α_V to lists, $\alpha_V^* : V_G^* \rightarrow V_H^*$. The action on morphisms is given by,

$$\mathcal{F}(\alpha)(f) := \begin{cases} 1_{\mathcal{F}(\alpha)(X)} & \text{if } f = 1_X, \\ \mathcal{F}(\alpha)(X) \triangleright \alpha_E(f') \triangleleft \mathcal{F}(\alpha)(Y) & \text{if } f = X \triangleright f' \triangleleft Y \\ \mathcal{F}(\alpha)(g) \circledast \mathcal{F}(\alpha)(h) & \text{if } f = g \circledast h. \end{cases}$$

and this action is well-defined.

Proof. Well-definedness follows from the fact $\mathcal{FG}$ and $\mathcal{FH}$ are categories and that α preserves orthogonality by definition. The action on objects is a morphism of monoids, so to see that $\mathcal{F}(\alpha)$ is a strict premonoidal functor, it remains to check that it preserves whiskerings: this follows by a straightforward induction on f . ◀

It is clear that the mapping $\mathcal{F}$ preserves composition and identities, and thus,

► **Proposition 28.** The assignments of Definition 24 and Lemma 27 define a functor $\mathcal{F} : \text{DevGraph} \rightarrow \text{PreMon}$.

To define the free effectful category over an effectful graph, we simply combine the construction of the free monoidal category on a monoidal graph (Construction 48) with the construction of the free premonoidal category on a device graph (Definition 24), as follows.

► **Definition 29.** Let $\mathcal{G} : V \rightarrow C$ be an effectful graph. This generates the following effectful category $\mathcal{FG} : \mathcal{F}_\otimes V \rightarrow \mathcal{FC}$, which Proposition 30 checks is well-defined.

- The monoidal category is the free monoidal category $\mathcal{F}_\otimes V$ on the monoidal graph V ,
- the premonoidal category $\mathcal{FC}$ is the free premonoidal category on the device graph C ,
- $\mathcal{FG} : \mathcal{F}_\otimes V \rightarrow \mathcal{FC}$ is taken to be identity-on-objects, and is defined by structural induction on its arguments, following Construction 48. The interesting case is, for $g_1 : A_1 \rightarrow B_1$, $g_2 : A_2 \rightarrow B_2$,

$$\mathcal{FG}(g_1 \otimes g_2) := (\mathcal{FG}(g_1) \times A_2)(B_1 \times \mathcal{FG}(g_2)) = (A_1 \times \mathcal{FG}(g_2))(\mathcal{FG}(g_1) \times B_2).$$

For this equality to hold, we must have that $\mathcal{FG}(g_1)$ and $\mathcal{FG}(g_2)$ are orthogonal for all g_1 and g_2 in $\mathcal{F}_\otimes V$, since then we can conclude from Lemma 25 that they interchange. It suffices to show that $\mathcal{FG}(g)$ has no devices for all g in $\mathcal{F}_\otimes V$, which follows by structural induction (c.f. Construction 48).

► **Proposition 30.** Definition 29 has the structure of an effectful category.

Proof. By the definitions of device graph, free monoidal category and free premonoidal category, we have that $(\mathcal{F}C)_{\text{obj}} = (\mathcal{F}_{\otimes}V)_{\text{obj}} := V_{\text{obj}}^*$. We must check that $\mathcal{F}\mathcal{G} : \mathcal{F}_{\otimes}V \rightarrow \mathcal{F}C$ is a strict premonoidal functor with central image. $\mathcal{F}\mathcal{G}$ is identity-on-objects and so a monoid homomorphism. Furthermore it preserves left-whiskering since,

$$\mathcal{F}\mathcal{G}(A \otimes f) := (\mathcal{F}\mathcal{G}(\text{id}_A) \rtimes B)(A \ltimes \mathcal{F}\mathcal{G}(f)) = A \rtimes \mathcal{F}\mathcal{G}(f),$$

using the equations of strict premonoidal categories, and preserves right-whiskerings in a similar fashion. It follows from Lemma 25 that the image of $\mathcal{F}\mathcal{G}$ is central since the definition of device graph asks that $\text{dev}_G(v) := \emptyset$ for all $v \in V$, and so by Definition 24, $\text{dev}(\mathcal{F}\mathcal{G}(g)) = \emptyset$ for all $g \in \mathcal{F}_{\otimes}V$. $\blacktriangleleft$

► **Proposition 31.** *Let $(\alpha_0, \alpha) : (\mathcal{G} : V \rightarrow C) \rightarrow (\mathcal{H} : W \rightarrow D)$ be a morphism of effectful graphs. This generates a morphism of effectful categories $\mathcal{F}(\alpha_0, \alpha) : \mathcal{F}\mathcal{G} \rightarrow \mathcal{F}\mathcal{H}$ defined as follows:*

- *the underlying strict monoidal functor is $\mathcal{F}_{\otimes}\alpha_0 : \mathcal{F}_{\otimes}V \rightarrow \mathcal{F}_{\otimes}W$,*
- *the underlying strict premonoidal functor $\mathcal{F}\alpha : \mathcal{F}C \rightarrow \mathcal{F}D$ is that given by Lemma 27,*
- *and the required square commutes by functoriality.*

It is easily checked that this assignment on morphisms preserves identities and composition.

► **Proposition 32.** *The assignments of Proposition 30 and Proposition 31 extend to a functor $\mathcal{F} : \text{EffGraph} \rightarrow \text{EffCat}$.*

4 Interference: from effectful categories to effectful graphs

In this section we show that every effectful category has an underlying effectful graph, given by a generalization of the clique construction of Proposition 5. Rather than taking the elements of a monoid as the vertices, we take the vertices to be the morphisms of a premonoidal category.

► **Definition 33.** *The interference graph of a premonoidal category $\mathbb{C}$, denoted by $\frac{1}{2}\mathbb{C}$, is the graph whose vertices are the arrows of $\mathbb{C}$, with an edge (f, g) if and only if f and g interfere in $\mathbb{C}$ (denoted $f \frac{1}{2} g$): that is, they do not interchange.*

► **Definition 34.** *Let $\mathbb{C}$ be a strict premonoidal category. We define its underlying device graph $\mathcal{U}(\mathbb{C})$ as follows:*

- *For the underlying monoidal graph we take $V_{\mathcal{U}(\mathbb{C})} := \mathbb{C}_{\text{obj}}$, and for every pair of lists $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ of objects of $\mathbb{C}_{\text{obj}}$, we take a set of arrows*

$$\mathcal{U}(\mathbb{C})(X_1, \dots, X_n; Y_1, \dots, Y_m) := \mathbb{C}(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m).$$

That is, there is an arrow $X_1, \dots, X_n \rightarrow Y_1, \dots, Y_m$ in $\mathcal{U}(\mathbb{C})$ for every morphism $X_1 \otimes \dots \otimes X_n \rightarrow Y_1 \otimes \dots \otimes Y_m$ in $\mathbb{C}$. Since each X_i and Y_j is an object of $\mathbb{C}$, whose set of objects already forms a monoid with operation denoted $\otimes$, a morphism $f : X_1 \otimes \dots \otimes X_n \rightarrow Y_1 \otimes \dots \otimes Y_m$ is included both in $\mathcal{U}(\mathbb{C})(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m)$ and $\mathcal{U}(\mathbb{C})(X_1, \dots, X_n; Y_1, \dots, Y_m)$. The complete set of arrows $E_{\mathcal{U}(\mathbb{C})}$ is the coproduct of all such sets, over every pair of lists of objects in $\mathbb{C}$.

- *The set of devices, $\text{cliques}(\frac{1}{2}\mathbb{C})$, is given by the set of non-trivial maximal cliques in the interference graph, $\frac{1}{2}\mathbb{C}$. A clique is trivial just when it is a singleton. A clique is maximal just in case any morphism that interferes with every morphism of the clique is in the clique. The device assignment, $\text{dev} : E_{\mathcal{U}(\mathbb{C})} \rightarrow \mathcal{P}(\text{cliques}(\frac{1}{2}\mathbb{C}))$, assigns an arrow to the subset of non-trivial maximal cliques to which it belongs.*

► **Lemma 35.** *For $F : \mathbb{X} \rightarrow \mathbb{Y}$ a strict premonoidal functor, we have that $f \parallel g$ implies $F(f) \parallel F(g)$, and that $F(f) \not\parallel F(g)$ implies $f \not\parallel g$.*

Proof. Immediate. ◀

► **Proposition 36.** *For every strict premonoidal functor $F : \mathbb{C} \rightarrow \mathbb{D}$, there is a morphism of device graphs $\mathcal{U}(F) : \mathcal{U}(\mathbb{C}) \rightarrow \mathcal{U}(\mathbb{D})$ given by the action of F .*

Proof. That this defines a morphism of the underlying monoidal graphs follows from the fact that a functor preserves sources and targets. We show that orthogonality of morphisms is preserved by proving the contrapositive, $\mathcal{U}(F)(f) \not\parallel \mathcal{U}(F)(g) \implies f \not\parallel g$. Suppose that $C \in \text{dev}_{\mathcal{U}(\mathbb{D})}(\mathcal{U}(F)(f)) \cap \text{dev}_{\mathcal{U}(\mathbb{D})}(\mathcal{U}(F)(g))$ for some $C \in \text{cliques}(\not\parallel \mathbb{D})$. Then $\mathcal{U}(F)(f) = F(f) \in C$ and $\mathcal{U}(F)(g) = F(g) \in C$, and we have $F(f) \not\parallel F(g)$. Then Lemma 35 gives $f \not\parallel g$, which means $f, g \in C'$ for some $C' \in \text{cliques}(\not\parallel \mathbb{C})$, in which case $C' \in \text{dev}_{\mathcal{U}(\mathbb{C})}(f) \cap \text{dev}_{\mathcal{U}(\mathbb{C})}(g)$, that is, $f \not\parallel g$, which is what we wanted to show. ◀

► **Lemma 37.** *The assignments of Definition 34 and Proposition 36 extend to a functor $\mathcal{U} : \text{PremonCat} \rightarrow \text{DevGraph}$.*

We can now define a functor $\mathcal{U} : \text{EffCat} \rightarrow \text{EffGraph}$. We begin with the action on effectful categories.

► **Proposition 38.** *Let $\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}$ be an effectful category. Then $\mathcal{U}(\mathcal{E}) : \mathcal{U}_{\otimes}(\mathbb{V}) \rightarrow \mathcal{U}(\mathbb{C})$ is the effectful graph with*

- monoidal graph $\mathcal{U}_{\otimes}(\mathbb{V})$ (Definition 50),
- device graph $\mathcal{U}(\mathbb{C})$,
- morphism of monoidal graphs $\mathcal{U}(\mathcal{E}) : \mathcal{U}_{\otimes}(\mathbb{V}) \rightarrow |\mathcal{U}(\mathbb{C})|$ defined to be identity-on-objects, and $f \mapsto \mathcal{E}(f)$ on arrows.

To see that this is well-defined, first note that by definition, $\mathcal{U}(\mathbb{C})$ has the same set of objects as $\mathcal{U}_{\otimes}(\mathbb{V})$. It remains to show that $\text{dev}(\mathcal{U}(\mathcal{E})(v)) = \emptyset$ for all arrows v in $\mathcal{U}_{\otimes}(\mathbb{V})$. By definition of $\mathcal{U}(\mathcal{E})$, we have $\mathcal{U}(\mathcal{E})(v) = \mathcal{E}(v)$. Also, by definition $\mathcal{E} : \mathbb{V} \rightarrow \mathbb{C}$ has central image, and so $\mathcal{E}(v)$ must constitute a trivial clique in $\not\parallel \mathbb{C}$.

It follows straightforwardly by combining Definition 50 and Proposition 38, that every morphism of effectful categories has an underlying morphism of effectful graphs, and furthermore that these assignments extend to a functor:

► **Proposition 39.** *Let $\mathcal{G} : \mathbb{V} \rightarrow \mathbb{C}$ and $\mathcal{H} : \mathbb{W} \rightarrow \mathbb{D}$ be effectful categories and $(\alpha_0, \alpha) : (\mathcal{G} : \mathbb{V} \rightarrow \mathbb{C}) \rightarrow (\mathcal{H} : \mathbb{W} \rightarrow \mathbb{D})$ a morphism of effectful categories. Then, there is an underlying morphism of effectful graphs, $\mathcal{U}(\alpha_0, \alpha) := (\mathcal{U}_{\otimes}(\alpha_0), \mathcal{U}(\alpha))$, where $\mathcal{U}_{\otimes}$ is the underlying morphism of monoidal graphs.*

► **Proposition 40.** *The assignments of Propositions 38 and 39 extend to a functor $\mathcal{U} : \text{EffCat} \rightarrow \text{EffGraph}$.*

Finally, we show that our free and forgetful functors form an adjunction. Proofs can be found in Appendix A.

► **Theorem 41.** *The free construction of premonoidal categories over a device graph (Definition 24) is left adjoint to forming the underlying device graph of a premonoidal category (Definition 34). That is, $\mathcal{F} \dashv \mathcal{U} : \text{PremonCat} \rightleftharpoons \text{DevGraph}$.*

► **Corollary 42.** *The adjunction of Theorem 41 extends to an adjunction between effectful graphs and effectful categories. That is, $\mathcal{F} \dashv \mathcal{U} : \text{EffCat} \rightleftharpoons \text{EffGraph}$.*

5 Commuting tensor product of free effectful categories

In their abstract investigation of the notion of *commutativity*, Garner and López Franco deduce the existence of a *commuting tensor product* of effectful categories [10, §9.4]. In this section, we show how our construction of free effectful categories from effectful graphs gives rise to a simple, concrete construction of this commuting tensor product.

Roughly speaking, given two free effectful categories generated by effectful graphs over the same monoidal graph of chosen pure actions, their commuting tensor product is given by taking the free effectful category over a sort of “disjoint union” of the two underlying effectful graphs. In particular, the set of devices is a disjoint union, and so actions from one graph are forced to interchange with those from the other. However, the set of resources is left unchanged, and so the actions from each graph may share resources when they are composed into resourceful traces, as in Figure 1 (see also Example 44, below).

► **Construction 43.** Given effectful graphs $\mathcal{G} : W \rightarrow C$ and $\mathcal{H} : W \rightarrow D$ over the same monoidal graph W , we construct their commuting tensor product $\mathcal{G} \odot \mathcal{H} : W \rightarrow C \odot D$ as follows. The monoidal graph of pure morphisms is simply W . The device graph $C \odot D$ has,

- objects $V_{C \odot D} := V_W = V_C = V_D$,
- arrows $E_{C \odot D} := E_C +_{E_W} E_D$, i.e., the following pushout of sets,

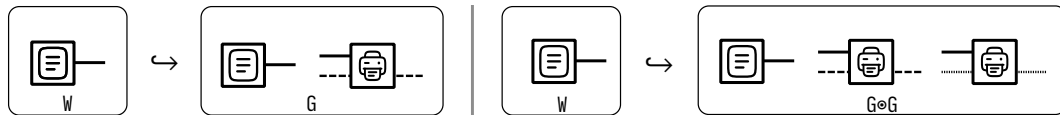
$$\begin{array}{ccc} E_W & \xrightarrow{\mathcal{H}_E} & E_D \\ \mathcal{G}_E \downarrow & & \downarrow R_E^{C,D} \\ E_C & \xrightarrow{L_E^{C,D}} & E_C +_{E_W} E_D \end{array}$$

- devices $\mathcal{D}_{C \odot D} := \mathcal{D}_C + \mathcal{D}_D$,
- device assignment $\text{dev}_{C \odot D} : E_C +_{E_W} E_D \rightarrow \mathcal{P}(\mathcal{D}_{C \odot D})$ the unique map induced by the universal property of the pushout via the assignments dev_C and dev_D ,

$$\begin{array}{ccccc} E_W & \xrightarrow{\mathcal{H}_E} & E_D & \xrightarrow{\text{dev}_D} & \mathcal{P}(\mathcal{D}_D) \\ \mathcal{G}_E \downarrow & & \downarrow R_E^{C,D} & & \downarrow \mathcal{P}(\text{inr}) \\ E_C & \xrightarrow{L_E^{C,D}} & E_C +_{E_W} E_D & & \mathcal{P}(\mathcal{D}_C + \mathcal{D}_D) \\ \text{dev}_C \downarrow & & \downarrow \text{dev}_{C \odot D} & \dashrightarrow & \parallel \\ \mathcal{P}(\mathcal{D}_C) & \xrightarrow{\mathcal{P}(\text{inl})} & \mathcal{P}(\mathcal{D}_C + \mathcal{D}_D) & \xlongequal{\quad} & \mathcal{P}(\mathcal{D}_{C \odot D}) \end{array}$$

- $\mathcal{G} \odot \mathcal{H} : W \rightarrow C \odot D$ is identity-on-objects and $L_E^{C,D} \circ \mathcal{G}_E (= R_E^{C,D} \circ \mathcal{H}_E)$ on arrows.

► **Example 44.** A simple example is given by our printer example from Section 1, reproduced in Figure 8 (left), with the chosen monoidal graph of pure actions W made explicit: in this case, it comprises the only pure action, namely the “document” generator. The commuting tensor product of this effectful graph with itself is the effectful graph in Figure 8 (right), in which we have two distinct printers.



■ **Figure 8** (Left) Effectful graph corresponding to the simple “theory of a printer”. (Right) Commuting tensor product of this effectful graph with itself: the theory of two printers that may be used in parallel.

Our next goal is to show that the free effectful category over a commuting tensor product of effectful graphs is isomorphic to the commuting tensor product of the free effectful categories over those graphs. First, we recall the definition of commuting tensor product of effectful categories given by Garner and López Franco [10, §9.4], see also [6, Proposition 4.6.4]. The definition builds on the auxiliary notion of commuting cospan.

► **Definition 45** (Commuting cospan). *A cospan of effectful categories*

$$F : (\mathcal{A} : \mathbb{V} \rightarrow \mathbb{A}) \rightarrow (\mathcal{K} : \mathbb{V} \rightarrow \mathbb{K}) \leftarrow (\mathcal{B} : \mathbb{V} \rightarrow \mathbb{B}) : G,$$

over the same monoidal category $\mathbb{V}$, is said to be commuting when, for every $f : A \rightarrow A'$ in $\mathbb{A}$ and $g : B \rightarrow B'$ in $\mathbb{B}$, the following diagram commutes in $\mathbb{K}$:

$$\begin{array}{ccc} A \otimes B & \xrightarrow{F(f) \times B} & A' \otimes B \\ A \times G(g) \downarrow & & \downarrow A' \times G(g) \\ A \otimes B' & \xrightarrow{F(f) \times B'} & A' \otimes B' \end{array}$$

► **Definition 46.** Let $\mathcal{A}$ and $\mathcal{B}$ be effectful categories over a monoidal category $\mathbb{V}$. The commuting tensor product of $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{A} \odot \mathcal{B}$, is defined to be the apex of the initial commuting cospan $L : \mathcal{A} \rightarrow \mathcal{A} \odot \mathcal{B} \leftarrow \mathcal{B} : R$ of effectful categories over $\mathbb{V}$. Explicitly, for any other commuting cospan, $F : \mathcal{A} \rightarrow \mathcal{K} \leftarrow \mathcal{B} : G$ of effectful categories over $\mathbb{V}$, there exists a unique morphism of effectful categories $\mathcal{A} \odot \mathcal{B} \rightarrow \mathcal{K}$ such that:

$$\begin{array}{ccccc} & & \mathcal{A} \odot \mathcal{B} & & \\ & \nearrow L & \downarrow & \nwarrow R & \\ \mathcal{A} & \xrightarrow{F} & \mathcal{K} & \xleftarrow{G} & \mathcal{B} \end{array}$$

► **Theorem 47.** Given effectful graphs $\mathcal{G} : W \rightarrow C$ and $\mathcal{H} : W \rightarrow D$ over the same monoidal graph W , $\mathcal{F}(\mathcal{G} \odot \mathcal{H}) \cong \mathcal{F}\mathcal{G} \odot \mathcal{F}\mathcal{H}$.

Proof. It suffices to show that $\mathcal{F}(\mathcal{G} \odot \mathcal{H})$ is the apex of the initial commuting cospan of effectful categories over the free monoidal category on W . In particular, we will show that:

$$\mathcal{F}\mathcal{G} \xrightarrow{\mathcal{F}(L^{\mathcal{G}, \mathcal{H}})} \mathcal{F}(\mathcal{G} \odot \mathcal{H}) \xleftarrow{\mathcal{F}(R^{\mathcal{G}, \mathcal{H}})} \mathcal{F}\mathcal{H}$$

is an initial commuting cospan, where $L^{\mathcal{G}, \mathcal{H}} : \mathcal{G} \rightarrow \mathcal{G} \odot \mathcal{H}$ is the morphism of effectful graphs that is identity on objects and pure generators, and given by the left pushout injection $L_E^{C,D}$ on generators (Construction 43), and similarly for $R^{\mathcal{G}, \mathcal{H}} : \mathcal{H} \rightarrow \mathcal{G} \odot \mathcal{H}$. This cospan is commuting because of the way in which $\text{dev}_{C \odot D}$ is constructed using the universal property of the pushout given by $L_E^{C,D}$ and $R_E^{C,D}$. In particular, we have by construction that for any $x \in E_C$ and $y \in E_D$, $L_E^{C,D}(x) \perp_{C \odot D} R_E^{C,D}(y)$. It follows that $\mathcal{F}(L^{\mathcal{G}, \mathcal{H}})(f) \perp_{\mathcal{F}(\mathcal{G} \odot \mathcal{H})} \mathcal{F}(R^{\mathcal{G}, \mathcal{H}})(g)$ for any morphisms f in $\mathcal{F}(\mathcal{G})$ and g in $\mathcal{F}(\mathcal{H})$, from which it follows in turn that the cospan in question is commuting.

To see that it is initial, suppose that we have another commuting cospan of effectful categories over $\mathcal{F}_\otimes W$, call it $\mathcal{F}\mathcal{G} \xrightarrow{P} (\mathcal{K} : \mathcal{F}_\otimes W \rightarrow \mathcal{K}) \xleftarrow{Q} \mathcal{F}\mathcal{H}$. We obtain a function

$(P \odot Q)_E : E_{C \odot D} \rightarrow E_{\mathcal{U}(\mathcal{K})}$ by the universal property of the pushout:

$$\begin{array}{ccccc}
 E_W & \xrightarrow{\mathcal{H}_E} & E_D & \xrightarrow{(\eta_D)_E} & E_{\mathcal{U}(\mathcal{F}(D))} \\
 \mathcal{G}_E \downarrow & & \downarrow R_E^{C,D} & & \downarrow \mathcal{U}(Q)_E \\
 E_C & \xrightarrow{L_E^{C,D}} & E_{C \odot D} & \xrightarrow{(P \odot Q)_E} & E_{\mathcal{U}(\mathcal{K})} \\
 (\eta_C)_E \downarrow & & & & \downarrow \mathcal{U}(P)_E \\
 E_{\mathcal{U}(\mathcal{F}(C))} & \xrightarrow{\mathcal{U}(P)_E} & & & E_{\mathcal{U}(\mathcal{K})}
 \end{array}$$

This mapping on generators of the device graphs defines an identity-on-objects morphism of effectful graphs $P \odot Q : \mathcal{G} \odot \mathcal{H} \rightarrow \mathcal{U}(\mathcal{K})$, where the fact that P and Q are a commuting cospan ensures that $f \perp_{\mathcal{G} \odot \mathcal{H}} g \Rightarrow (P \odot Q)(f) \perp_{\mathcal{U}(\mathcal{K})} (P \odot Q)(g)$. Using the bijection of hom-sets given by Corollary 42, we obtain a morphism $(P \odot Q)^\# : \mathcal{F}(\mathcal{G} \odot \mathcal{H}) \rightarrow \mathcal{K}$ of EffCat such that:

$$\begin{array}{ccccc}
 \mathcal{F}G & \xrightarrow{\mathcal{F}(L^{G,H})} & \mathcal{F}(\mathcal{G} \odot \mathcal{H}) & \xleftarrow{\mathcal{F}(R^{G,H})} & \mathcal{F}H \\
 & \searrow P & \downarrow (P \odot Q)^\# & \swarrow Q & \\
 & & \mathcal{K} & &
 \end{array}$$

Any other such morphism, transported back across the adjunction, induces another filler for our pushout diagram, and must therefore be equal to $(P \odot Q)^\#$. $\blacktriangleleft$

6 Conclusion and further work

We have shown how morphisms of free effectful categories can be seen as generalizations of Mazurkiewicz traces, by introducing a new notion of generating data for effectful categories, namely *effectful graphs*. This point of view also leads to a simple presentation of the commuting tensor product of free effectful categories. In this paper, we have not given a treatment of presentations involving equations but this would be a natural next step, and would allow us to express richer descriptions of concurrent systems, such as commuting tensor products of global state [22].

Prior work has generalized Mazurkiewicz traces along different lines, such as to contextual traces [2] or semicommutations [3], so these might be investigated in light of this paper. For example, semicommutations generalize Mazurkiewicz traces by allowing commutations to be *directed*, and this may correspond to a notion of *order-enriched effectful category*.

Since effectful categories are equivalent to strong promonads [21, 10], recasting our concepts from that point of view would bring our work adjacent to the literature on combining monads [11], whose relation to the commuting tensor product should be investigated. This may also suggest appropriate generalizations. For example, we might be able to see device information as corresponding to a grading of a strong promonad.

String diagrams bearing some resemblance to resourceful traces have been used informally by Mellies in a treatment of the local state monad [17]. The local state monad arises as a combination of global state monads, linked by a theory of allocation: connections to our commuting tensor product should be investigated. Similarly, Barrett, Heijltjes, and McCusker [1] have made informal use of string diagrams bearing resemblance to resourceful traces, in which devices stand for distinct effects in an effectful λ -calculus. These instances suggest

that it would be worthwhile to formalize the diagrammatic point of view. Sobociński and the third author [22] proved that string diagrams over effectful graphs with a single global device do indeed give free premonoidal categories. However, since it is possible to construct effectful categories whose interference graphs contain an infinite number of maximal cliques, making the string-diagrammatic construction formal in our setting would require restricting to an appropriate subcategory of finitary effectful graphs. With this restriction, we see no reason why string diagrams for effectful categories should not extend to our setting.

References

- 1 Chris Barrett, Willem Heijltjes, and Guy McCusker. The Functional Machine Calculus II: Semantics. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2023/17471>, doi:10.4230/LIPIcs.CSL.2023.10.
- 2 Christian Choffrut and Robert Mercas. Contextual partial commutations. *Discrete Mathematics & Theoretical Computer Science*, 12, 2010.
- 3 Mireille Clerbout and Michel Latteux. Semi-commutations. *Information and computation*, 73(1):59–74, 1987.
- 4 Bob Coecke, Tobias Fritz, and Robert W Spekkens. A mathematical theory of resources. *Information and Computation*, 250:59–86, 2016.
- 5 Volker Diekert and Grzegorz Rozenberg. *The Book of Traces*. World Scientific, 1995. URL: <https://www.worldscientific.com/doi/abs/10.1142/2563>, doi:10.1142/2563.
- 6 Matthew Earnshaw. *Languages of String Diagrams*. PhD thesis, Tallinn University of Technology, 2025. URL: <https://digikogu.taltech.ee/et/Item/30c6f3d0-9655-483d-bd8d-46a5a68bcade>, doi:10.23658/TALTECH.2/2025.
- 7 Matthew Earnshaw and Pawel Sobociński. String Diagrammatic Trace Theory. In *MFCS*, volume 272, 2023. doi:10.4230/LIPIcs.MFCS.2023.43.
- 8 Dominique Foata and Pierre Cartier. Problèmes combinatoires de commutation et réarrangements. *Lecture notes in mathematics*, 85, 1969.
- 9 P. Freyd. Algebra valued functors in general and tensor products in particular. *Colloquium Mathematicum*, 14(1):89–106, 1966. URL: <http://dx.doi.org/10.4064/cm-14-1-89-106>, doi:10.4064/cm-14-1-89-106.
- 10 Richard Garner and Ignacio López Franco. Commutativity. *Journal of Pure and Applied Algebra*, 220(5):1707–1751, 2016.
- 11 Martin Hyland, Gordon Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical computer science*, 357(1-3):70–99, 2006.
- 12 Alan Jeffrey. Premonoidal categories and a graphical view of programs. *Preprint*, 1997.
- 13 André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. URL: <https://www.sciencedirect.com/science/article/pii/000187089190003P>, doi:10.1016/0001-8708(91)90003-P.
- 14 Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Information and Computation*, 185(2):182–210, 2003. URL: <https://www.sciencedirect.com/science/article/pii/S0890540103000889>, doi:10.1016/S0890-5401(03)00088-9.
- 15 Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. *DAIMI Report Series*, 6(78), July 1977. URL: <http://dx.doi.org/10.7146/dpb.v6i78.7691>, doi:10.7146/dpb.v6i78.7691.
- 16 Antoni Mazurkiewicz. Basic notions of trace theory. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 285–363, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

- 17 Paul-André Melliès. Local states in string diagrams. In *Rewriting and Typed Lambda Calculi: Joint International Conference*, pages 334–348. Springer, 2014.
- 18 John Power. Premonoidal categories as categories with algebraic structure. *Theoretical Computer Science*, 278(1):303–321, 2002. Mathematical Foundations of Programming Semantics 1996. URL: <https://www.sciencedirect.com/science/article/pii/S0304397500003406>, doi:10.1016/S0304-3975(00)00340-6.
- 19 John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Math. Struct. Comput. Sci.*, 7(5):453–468, 1997. doi:10.1017/S0960129597002375.
- 20 John Power and Hayo Thielecke. Closed freyd- and κ -categories. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming*, pages 625–634, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
- 21 Mario Román. Promonads and string diagrams for effectful categories. In *Applied Category Theory*, volume 380 of *Electronic Proceedings in Theoretical Computer Science*, pages 344–361. Open Publishing Association, 2023. doi:10.4204/EPTCS.380.20.
- 22 Mario Román and Paweł Sobociński. String diagrams for premonoidal categories. *Logical Methods in Computer Science*, Volume 21, Issue 2, Apr 2025. URL: <https://lmcs.episciences.org/13384>, doi:10.46298/lmcs-21(2:9)2025.
- 23 Sam Staton and Paul Blain Levy. Universal properties of impure programming languages. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, page 179–192, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2429069.2429091.
- 24 Wiesław Zielonka. Notes on finite asynchronous automata. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 21(2):99–135, 1987. URL: <http://www.numdam.org/item/ITA%5F1987%5F%5F21%5F2%5F99%5F0/>.

A

 Omitted details

► **Lemma 25.** *Consider the inductive extension of the device assignment of a device graph G to morphisms of $\mathcal{FG}$ (Definition 24),*

$$\text{dev}_{\mathcal{FG}}(f) := \begin{cases} \emptyset & \text{if } f = 1_A, \\ \text{dev}_G(f') & \text{if } f = X \triangleright f' \triangleleft Y, \\ \text{dev}_{\mathcal{FG}}(g) \cup \text{dev}_{\mathcal{FG}}(h) & \text{if } f = g \circ h. \end{cases}$$

Let f, g be two morphisms of $\mathcal{FG}$. If f and g are *orthogonal*, $\text{dev}_{\mathcal{FG}}(f) \cap \text{dev}_{\mathcal{FG}}(g) = \emptyset$, then $f \parallel g$ and $g \parallel f$.

Proof. By structural induction on f and g . The cases where one of f and g are an identity or both morphisms are whiskered generators are straightforward by the equations of premonoidal categories, or by the imposed interchange equations for whiskered generators. Let $f = f_1 \circ f_2$, then by assumption we obtain $\text{dev}_{\mathcal{FG}}(f_1) \cap \text{dev}_{\mathcal{FG}}(g) = \text{dev}_{\mathcal{FG}}(f_2) \cap \text{dev}_{\mathcal{FG}}(g) = \emptyset$, and by hypothesis $f_1 \parallel g$ and $f_2 \parallel g$. Say $f_1 : A \rightarrow B, f_2 : B \rightarrow C, g : D \rightarrow E$. We need to show $((f_1 \circ f_2) \times D) \circ (C \times g) = (A \times g) \circ ((f_1 \circ f_2) \times E)$. By the functoriality of $\times$ and associativity of $\circ$, the left-hand side is equal to $(f_1 \times D) \circ ((f_2 \times D) \circ (C \times g))$. Using the two induction hypotheses, this becomes $(A \times g) \circ ((f_1 \times E) \circ (f_2 \times E))$ which is finally equal to the right-hand side by the functoriality of $\times$. The argument for $g \parallel f$ is symmetric, as well as the case when g is a composite. ◀

► **Construction 48** (*Free strict monoidal category*). For a monoidal graph G , we construct a strict monoidal category $\mathcal{F}_{\otimes}G$ on G as follows. The underlying category of $\mathcal{F}_{\otimes}G$ has:

- set of objects V_G^* , the free monoid on V_G ,
- set of morphisms constructed according to the following rules:

$$\begin{array}{c} \text{GEN} \\ \frac{f : x \rightarrow y \in G}{f : x \rightarrow y} \end{array} \quad \begin{array}{c} \text{ID} \\ \frac{A \in V_G^*}{1_A : A \rightarrow A} \end{array} \quad \begin{array}{c} \text{COMP} \\ \frac{f : A \rightarrow B \quad g : B \rightarrow C}{fg : A \rightarrow C} \end{array}$$

$$\begin{array}{c} \text{TENSOR} \\ \frac{f_1 : A_1 \rightarrow B_1 \quad f_2 : A_2 \rightarrow B_2}{f_1 \otimes f_2 : A_1 \otimes A_2 \rightarrow B_1 \otimes B_2} \end{array}$$

subject to the equations,

$$\begin{aligned} (fg)h &= f(gh) & 1_A f &= f = f 1_B & (1_A \otimes 1_B) &= 1_{A \otimes B} \\ (f_1 \otimes f_2)(g_1 \otimes g_2) &= (f_1 g_1) \otimes (f_2 g_2) & (f \otimes g) \otimes h &= f \otimes (g \otimes h) \end{aligned}$$

- **composition** and **identities** are given by the corresponding inference rules.

The first two equations ensure that this data forms a category and the remaining equations ensure that the evident monoidal product is functorial.

Similarly, the free strict monoidal functor on a morphism of monoidal graphs is given by the evident inductive extension.

► **Proposition 49.** The construction of $\mathcal{F}_{\otimes}$ defines a functor $\mathcal{F}_{\otimes} : \text{MonGraph} \rightarrow \text{MonCat}$.

► **Definition 50.** The underlying monoidal graph $\mathcal{U}_{\otimes}M$ of a monoidal category M is defined to have set of objects the set of objects of M , and set of arrows,

$$\coprod_{\substack{(X_1 \in M_{\text{obj}}, \dots, X_n \in M_{\text{obj}}) \\ (Y_1 \in M_{\text{obj}}, \dots, Y_m \in M_{\text{obj}})}} M(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m)$$

with the obvious assignments of sources and targets of arrows. For a strict monoidal functor $F : M \rightarrow N$, its underlying morphism of polygraphs $\mathcal{U}_{\otimes}(F)$ is given by the action of F on objects and arrows.

► **Proposition 51.** *Definition 50 defines a functor $\mathcal{U}_{\otimes} : \text{MonCat} \rightarrow \text{MonGraph}$.*

► **Proposition 52.** *There is an adjunction $\mathcal{F}_{\otimes} \dashv \mathcal{U}_{\otimes} : \text{MonCat} \rightleftarrows \text{MonGraph}$.*

Proof. A left adjoint to $\mathcal{U}_{\otimes}$ is constructed via string diagrams by Joyal and Street [13]. ◀

► **Theorem 41.** *The free construction of premonoidal categories over a device graph (Definition 24) is left adjoint to forming the underlying device graph of a premonoidal category (Definition 34). That is, $\mathcal{F} \dashv \mathcal{U} : \text{PremonCat} \rightleftarrows \text{DevGraph}$.*

We begin constructing the unit, $\eta : 1_{\text{DevGraph}} \rightarrow \mathcal{U} \circ \mathcal{F}$. A component $\eta_G : G \rightarrow \mathcal{U}(\mathcal{F}(G))$ acts on objects by inclusion as a singleton list, which we again denote simply as A ,

$$(\eta_G)_V : V_G \rightarrow V_{\mathcal{U}(\mathcal{F}(G))} : A \mapsto A.$$

On arrows, recall that by definition,

$$E_{\mathcal{U}(\mathcal{F}(G))} = \coprod_{\substack{(X_1 \in \mathcal{F}(G)_{\text{obj}}, \dots, X_n \in \mathcal{F}(G)_{\text{obj}}) \\ (Y_1 \in \mathcal{F}(G)_{\text{obj}}, \dots, Y_m \in \mathcal{F}(G)_{\text{obj}})}} \mathcal{F}G(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m),$$

where $\otimes$ is list concatenation. Let A_i, B_j be elements of V_G , and $f : A_1, \dots, A_n \rightarrow B_1, \dots, B_m$ a morphism in $\mathcal{F}(G)$. Let us denote the coproduct injection into the component $(A_1, \dots, A_n), (B_1, \dots, B_m)$ of $E_{\mathcal{U}(\mathcal{F}(G))}$ by in . Then we define,

$$(\eta_G)_E : E_G \rightarrow E_{\mathcal{U}(\mathcal{F}(G))} : f \mapsto \text{in} \triangleright f \triangleleft,$$

where $\triangleright f \triangleleft$ denotes the primitive left-right whiskering of f by the empty list, as in Definition 24. It is easy to check that this is a morphism of the underlying monoidal graphs. For the preservation of orthogonality, if $f \perp_G g$ then by definition $\triangleright f \triangleleft \parallel \triangleright g \triangleleft$ and $\triangleright g \triangleleft \parallel \triangleright f \triangleleft$ in $\mathcal{F}(G)$. Therefore, $\triangleright f \triangleleft$ and $\triangleright g \triangleleft$ are not connected in the interference graph of $\mathcal{F}(G)$. Now, two maps in $\mathcal{U}(\mathcal{F}(G))$ will be orthogonal as long as they do not have a device in common, but they were to share a device, they would belong to a common maximal clique and thus be connected. This is a contradiction, so they must be orthogonal, $\eta_G(f) \perp_{\mathcal{U}(\mathcal{F}(G))} \eta_G(g)$.

It remains to show that η is natural. We must show that for any $\alpha : G \rightarrow H$ in DevGraph we have:

$$\begin{array}{ccc} G & \xrightarrow{\eta_G} & \mathcal{U}(\mathcal{F}(G)) \\ \alpha \downarrow & & \downarrow \mathcal{U}(\mathcal{F}(\alpha)) \\ H & \xrightarrow{\eta_H} & \mathcal{U}(\mathcal{F}(H)) \end{array}$$

We proceed by showing that the two above composite device graph morphisms have equal object mappings, and equal arrow mappings. First note $\mathcal{U}(\mathcal{F}(\alpha))_V = \alpha_V^*$ by definition, so we have,

$$\begin{aligned} ((\eta_H)_V \circ \alpha_V)A &= \alpha_V(A) = \alpha_V^*(A) \\ &= (\alpha_V^* \circ (\eta_G)_V)A = (\mathcal{U}(\mathcal{F}(\alpha))_V \circ (\eta_G)_V)A \end{aligned}$$

for all $A \in V_G$, as required. For the arrows, recall that by Lemma 27, $\mathcal{F}(\alpha)(\triangleright f \triangleleft) = \triangleright \alpha_E(f) \triangleleft$, which gives:

$$\begin{aligned} ((\eta_H)_E \circ \alpha_E)f &= (\eta_H)_E(\alpha_E(f)) = \text{in} \triangleright \alpha_E(f) \triangleleft = \text{in} F(\alpha)(\triangleright f \triangleleft) \\ &= \mathcal{U}(F(\alpha))_E(\text{in} \triangleright f \triangleleft) = \mathcal{U}(F(\alpha))_E(\eta_G f) = (\mathcal{U}(F(\alpha))_E \circ (\eta_G))f \end{aligned}$$

establishing the naturality of $\eta : 1_{\text{DevGraph}} \rightarrow \mathcal{U} \circ \mathcal{F}$.

We now turn our attention to the counit $\varepsilon : \mathcal{F} \circ \mathcal{U} \rightarrow 1_{\text{PremonCat}}$, where $\varepsilon_{\mathbb{C}} : \mathcal{F}(\mathcal{U}(\mathbb{C})) \rightarrow \mathbb{C}$ has action on objects $(\varepsilon_{\mathbb{C}})(A_1, \dots, A_n) := A_1 \otimes \dots \otimes A_n$, where $\otimes$ is the monoidal operation on objects that exists in a premonoidal category. The action on morphisms is defined by structural recursion as follows:

$$\varepsilon_{\mathbb{C}}(f) = \begin{cases} 1_{\varepsilon_{\mathbb{C}}(A)} & \text{if } f = 1_A \\ \varepsilon_{\mathbb{C}}(g) \circ \varepsilon_{\mathbb{C}}(h) & \text{if } f = g \circ h \\ \varepsilon_{\mathbb{C}}(X) \triangleright f' \triangleleft \varepsilon_{\mathbb{C}}(Y) & \text{if } f = X \triangleright \text{in } f' \triangleleft Y, \end{cases}$$

where in denotes a coproduct injection. It is straightforward to check that this is well-defined, is a monoid morphism on objects, and preserves whiskerings. It remains to show that ε is natural. We must show that for any $F : \mathbb{C} \rightarrow \mathbb{D}$ in PremonCat we have:

$$\begin{array}{ccc} \mathcal{F}(\mathcal{U}(\mathbb{C})) & \xrightarrow{\varepsilon_{\mathbb{C}}} & \mathbb{C} \\ \mathcal{F}(\mathcal{U}(F)) \downarrow & & \downarrow F \\ \mathcal{F}(\mathcal{U}(\mathbb{D})) & \xrightarrow{\varepsilon_{\mathbb{D}}} & \mathbb{D} \end{array}$$

We proceed by showing that the two composite functors have equal object mappings and equal arrow mappings. For the mapping on objects we have,

$$\begin{aligned} (F \circ \varepsilon_{\mathbb{C}})(A_1, \dots, A_n) &= F(A_1 \otimes \dots \otimes A_n) \\ &= F(A_1) \otimes \dots \otimes F(A_n) = (\varepsilon_{\mathbb{D}} \circ \mathcal{F}(\mathcal{U}(F)))(A_1, \dots, A_n). \end{aligned}$$

For the arrow mappings, we proceed by induction on f . The interesting case is when $f = X \triangleright \text{in } f' \triangleleft Y$, for $\text{in } f'$ an arrow of $\mathcal{U}(\mathbb{C})$,

$$\begin{aligned} (F \circ \varepsilon_{\mathbb{C}})(X \triangleright \text{in } f' \triangleleft Y) &= F(\varepsilon_{\mathbb{C}}(X) \triangleright \text{in } f' \triangleleft \varepsilon_{\mathbb{C}}(Y)) = F(\varepsilon_{\mathbb{C}}(X)) \triangleright F(\text{in } f') \triangleleft F(\varepsilon_{\mathbb{C}}(Y)) \\ &= \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(X)) \triangleright F(\text{in } f') \triangleleft \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(Y)) \\ &= \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(X \triangleright \text{in } f' \triangleleft Y)) = (\varepsilon_{\mathbb{D}} \circ \mathcal{F}(\mathcal{U}(F)))(X \triangleright \text{in } f' \triangleleft Y). \end{aligned}$$

The other cases follow directly from the inductive hypothesis together with the fact that each composite is a strict premonoidal functor. It follows that $\varepsilon : \mathcal{F} \circ \mathcal{U} \rightarrow 1_{\text{PremonCat}}$ is natural.

Finally, to obtain an adjunction we must show the triangle identities:

$$\begin{array}{ccc} \mathcal{U} & \xrightarrow{\mathcal{U}\eta} & \mathcal{U} \circ \mathcal{F} \circ \mathcal{U} \\ & \searrow 1_{\mathcal{U}} & \downarrow \varepsilon_{\mathcal{U}} \\ & & \mathcal{U} \end{array} \quad \begin{array}{ccc} \mathcal{F} & \xrightarrow{\eta_{\mathcal{F}}} & \mathcal{F} \circ \mathcal{U} \circ \mathcal{F} \\ & \searrow 1_{\mathcal{F}} & \downarrow \mathcal{F}\varepsilon \\ & & \mathcal{F} \end{array}$$

For the first triangle, we have:

$$(\varepsilon_{\mathcal{U}} \circ \mathcal{U}\eta)_{\mathbb{C}}(f) = \varepsilon_{\mathcal{U}}(\mathcal{U}\eta_{\mathbb{C}}(f)) = \mathcal{U}(\varepsilon_{\mathbb{C}})(f) = f = 1_{\mathcal{U}(\mathbb{C})}(f),$$

from which we may conclude that the triangle in question commutes. For the second triangle, we proceed by induction on f . The interesting case is when f is a whiskered generating arrow, $f = X \triangleright \text{inf}' \triangleleft Y$,

$$\begin{aligned}
 (\mathcal{F}\varepsilon \circ \eta\mathcal{F})_G(X \triangleright \text{inf}' \triangleleft Y) &= \mathcal{F}\varepsilon_G(\eta\mathcal{F}_G(X \triangleright \text{inf}' \triangleleft Y)) \\
 &= \mathcal{F}\varepsilon_G(\mathcal{F}(\eta_G)(X \triangleright \text{inf}' \triangleleft Y)) \\
 &= \mathcal{F}\varepsilon_G(\mathcal{F}(\eta_G)(X) \triangleright \mathcal{F}(\eta_G)(\text{inf}') \triangleleft \mathcal{F}(\eta_G)(Y))) \\
 &= \mathcal{F}\varepsilon_G([X] \triangleright \eta_G(\text{inf}') \triangleleft [Y]) \\
 &= \varepsilon_{\mathcal{F}G}([X]) \triangleright \text{inf}' \triangleleft \varepsilon_{\mathcal{F}G}([Y]) \\
 &= X \triangleright \text{inf}' \triangleleft Y,
 \end{aligned}$$

where for $X = X_1, \dots, X_n$, we denote by $[X]$ the list $[X_1], \dots, [X_n]$. The case where f is 1_A follows from the (easily checked) fact that the claim holds for the object mapping, and the case of composition follows directly from the inductive hypothesis together with the fact that the composite is a strict premonoidal functor.

► **Corollary 42.** *The adjunction of Theorem 41 extends to an adjunction between effectful graphs and effectful categories. That is, $\mathcal{F} \dashv \mathcal{U} : \text{EffCat} \rightleftharpoons \text{EffGraph}$.*

Proof. We shall make use of the adjunction of Theorem 41 and the adjunction between monoidal graphs and monoidal categories ([13]). Let $\mathcal{E} : V \rightarrow G$ be an effectful graph. We first define the components of the unit, namely a morphism of effectful graphs

$$\eta_{\mathcal{E}} : (\mathcal{E} : V \rightarrow G) \rightarrow (\mathcal{U} \circ \mathcal{F})(\mathcal{E} : V \rightarrow G),$$

by letting the morphism between the monoidal graphs be the unit of the adjunction between monoidal graphs and monoidal categories, $\eta_V^{\otimes} : V \rightarrow \mathcal{U}_{\otimes}\mathcal{F}_{\otimes}V$, and the morphism between the device graphs be given by the unit $\eta_G : G \rightarrow \mathcal{U}\mathcal{F}G$ of the adjunction constructed in Theorem 41. It is straightforward to verify that the required square commutes.

Let $\mathcal{C} : \mathbb{V} \rightarrow \mathbb{C}$ be an effectful category. For the components of the counit, we define a morphism of effectful categories,

$$\varepsilon_{\mathcal{C}} : (\mathcal{F} \circ \mathcal{U})\mathcal{C} \rightarrow \mathcal{C}$$

by letting the component between the monoidal categories be given by the counit of the adjunction between monoidal graphs and monoidal categories, $\varepsilon_{\mathbb{C}}^{\otimes} : \mathcal{F}_{\otimes}\mathcal{U}_{\otimes}\mathbb{C} \rightarrow \mathbb{C}$, and the morphism between premonoidal categories be given by the counit of the adjunction constructed in Theorem 41.

It remains to show that the unit and counit satisfy the triangle identities, which follows simply from the triangle identities of the two adjunctions involved. ◀