Resourceful Traces for Commuting Processes

Matthew Earnshaw
Tallinn University of Technology, Estonia
University of Tartu, Estonia

Chad Nester
University of Tartu, Estonia

Mario Román
University of Oxford, UK

Abstract

We show that, when the actions of a Mazurkiewicz trace are considered not merely as atomic but as 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 generalized 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 representation of morphisms in 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

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.28

Funding *Matthew Earnshaw*: Estonian Research Council grant PRG1210 and PRG2764.
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.

Acknowledgements The authors thank Niels Voorneveld for discussions, and the anonymous reviewers for their detailed comments and suggestions on the manuscript.

1 Introduction

Mazurkiewicz traces [5, 17] 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* [17, 25] 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" [16]. 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. In what follows, we shall refer to these simply as *resources*.

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 signature*. In our example, the effectful signature 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.

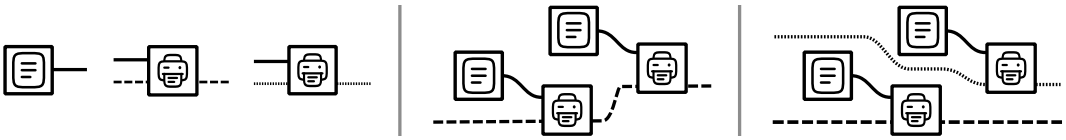


Figure 1 (Left) An effectful signature. (Center) Resourceful trace over the first two elements of the effectful signature, i.e. involving only one printer. (Right) Resourceful trace involving two printers: printings are now independent of one another.

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 signature assembles into an *effectful category* [23], also known as generalized Freyd categories [10, 21], 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 in free effectful categories. That is, we show that there is an adjunction between effectful signatures and effectful categories, generalizing the classic maximal cliques construction for Mazurkiewicz traces.

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 in free effectful categories with no resources (qua "indefinite noun phrases"). The construction of free effectful categories over effectful signatures with a single device was given by the third author, following Jeffrey [13, 22, 23]. 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 signatures 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 signatures* and *effectful categories*, via their underlying *device signatures* and *monoidal signatures*, generalizing the notion of distribution of an alphabet of actions. We give a construction of the free effectful category over an effectful signature, 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, Hoogeboom and Rozenberg [5]. For this section, we fix a set of actions Σ, and denote by Σ^* the monoid of words over 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 $a, b \in D$, $(a, b) \notin D$ 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 $\mathsf{dev} : \Sigma \to \mathscr{P}(\{1, ..., k\})$ for some $k \geqslant 1$. Distributions form a preorder Dist_Σ with $\mathsf{dev} \leqslant \mathsf{dev}'$ if and only if for all $a, b \in \Sigma$, $\mathsf{dev}(a) \cap \mathsf{dev}(b) = \varnothing \implies \mathsf{dev}'(a) \cap \mathsf{dev}'(b) = \varnothing$.*

Classically, $\mathsf{dev}(\sigma)$ is known as the set of *locations* of σ. In line with the terminology introduced in the following, we call $\mathsf{dev}(\sigma)$ the set of *devices* of σ. In terms of concurrency, we might consider $\mathsf{dev}(\sigma)$ 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.

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,

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

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].

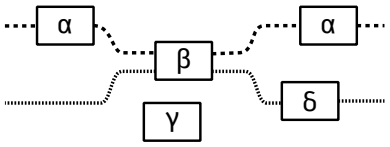


Figure 3 Graphical representation of a trace over the distribution in Figure 2. As an equivalence class of words in the trace monoid, this trace would include for example $\gamma\alpha\beta\alpha\delta$ and $\alpha\beta\gamma\delta\alpha$: independence is captured by *sliding* actions, while keeping 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.** ([7, §4.1]) *Every dependency relation D on Σ gives rise to a distribution of Σ by taking a bijection of $\{1, ..., k\}$ 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 (contravariant) Galois insertion,* cliques$\colon \mathsf{Dep}_\Sigma \hookrightarrow \mathsf{Dist}_\Sigma$.

Finally, we remark that any monoid has an underlying distribution, which can be found using the maximal cliques construction. A monoid M has a 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 signature. In an effectful signature, in addition to a set of devices, every action is equipped with input and output types of resources. In Section 3.3, we show that just as a distribution generates a free partially commutative monoid, effectful signatures generate free *effectful categories* [23], also known as *generalized Freyd categories* [10, 21].

3.1 Effectful signatures

Effectful signatures 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* [19, 15].

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 $M \to N$ 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 [24, §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 signature*:

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

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

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

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

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

It is easy to check that morphisms of monoidal signatures 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 signatures and their morphisms.*

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

▶ **Definition 8.** *A* device signature C *comprises*

- *a monoidal signature* $|C|$*,*
- *a set* $\mathcal{D}_C$ *of devices, and*
- *a function* $\mathsf{dev}_C : E_{|C|} \to \mathcal{P}(\mathcal{D}_C)$ *assigning each arrow in* $|C|$ *to a subset of the devices.*

We shall simply write dev *when the device signature* C *is clear from context.*

Figure 5 Left: monoidal signature with arrows $\alpha : X \to \varepsilon$, $\beta : XY \to X$. Right: device signature with the same underlying monoidal signature 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.

When a generator has a finite subset of devices, we can depict a device signature as on the right of Figure 5. Patterned strings denote devices (with arbitrary order of appearance). In contrast to resource strings, device strings must *thread through* the process: this enforces a definite ordering of dependent processes.

▶ **Example 9.** A device signature in which the underlying monoidal signature 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 signature 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 signature C are said to be* orthogonal*, denoted $f \perp_C g$ just when they share no devices, $\mathsf{dev}(f) \cap \mathsf{dev}(g) = \varnothing$. 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 signatures* must preserve orthogonality, that is

▶ **Definition 11.** *A* morphism of device signatures*, $\alpha : C \to D$, comprises*
- *a morphism of underlying monoidal signatures, $\alpha : |C| \to |D|$,*
- *such that for all arrows $f, g \in C$, if $f \perp_C g$ then $\alpha(f) \perp_D \alpha(g)$.*

Composition of these morphisms is given by composition of their underlying morphisms of monoidal signatures. It is easily checked that preservation of orthogonality is satisfied by a composite, and so device signatures and their morphisms form a category $\mathsf{DevGraph}$.

We now have the ingredients required to define *effectful signatures*.

▶ **Definition 12.** *An* effectful signature *$\mathcal{G} : V \to C$ comprises*
- *a monoidal signature V,*
- *a device signature C over the same set of objects as V,*
- *an identity-on-objects morphism of monoidal signatures $\mathcal{G} : V \to |C|$,*
- *such that $\mathsf{dev}(\mathcal{G}v) = \varnothing$ 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 signature*, we are referring to the arrows of the device signature C.

▶ **Example 13.** In the effectful signature in Figure 1, the monoidal signature comprises solely the "document" generator, while the device signature includes all three generators, with the morphism of monoidal signatures simply including the "document" generator.

▶ **Definition 14.** *A* morphism of effectful signatures *$(\alpha_0, \alpha) : (\mathcal{G} : V \to C) \to (\mathcal{H} : W \to D)$ comprises*
- *a morphism of monoidal signatures $\alpha_0 : V \to W$, and*
- *a morphism of device signatures $\alpha : C \to D$,*
- *such that the square of morphisms of monoidal signatures commutes, $\mathcal{G} \fatsemi \alpha = \alpha_0 \fatsemi \mathcal{H}$.*

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

▶ **Proposition 15.** *Effectful signatures and their morphisms form a category* EffGraph.

Note that effectful signatures generalize the *effectful polygraphs* of Sobociński and the third author [23], 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* [23], in that it builds in extra independence equations between processes.

3.2 Effectful categories

Effectful signatures generate *effectful categories*. An effectful category comprises a monoidal category $\mathbb{V}$, a premonoidal category $\mathbb{C}$, and an identity-on-objects premonoidal functor $\mathbb{V} \to \mathbb{C}$. 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 (pre)monoidal category is equivalent to a strict one [20].

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

▶ **Definition 16.** *A* strict monoidal category *is a category $\mathbb{V}$ equipped with*
- *a functor $\otimes : \mathbb{V} \times \mathbb{V} \to \mathbb{V}$, 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 arbitrary morphisms in parallel.

▶ **Definition 17.** *A* strict premonoidal category *is a category, $\mathbb{C}$, equipped with functors $(A \ltimes -) : \mathbb{C} \to \mathbb{C}$ (the "left-whiskering" by A) and $(- \rtimes A) : \mathbb{C} \to \mathbb{C}$ (the "right-whiskering" by A), for every object A of $\mathbb{C}$, and a distinguished object I (the "monoidal unit"), such that,*
- *$A \rtimes B = A \ltimes B$, 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 hold, stating coherence of left- and right-whiskering with each other and with the monoid structure*

$$(A \ltimes f) \rtimes B = A \ltimes (f \rtimes B) \qquad I \ltimes f = f = f \rtimes I$$
$$(A \otimes B) \ltimes f = A \ltimes (B \ltimes f) \qquad f \rtimes (A \otimes B) = (f \rtimes A) \rtimes B.$$

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 18.** *For morphisms $f : A \to B$ and $g : A' \to B'$ 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 $f \not\parallel g$, 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 19.** *Let $\mathbb{C}$ and $\mathbb{D}$ be strict premonoidal categories. A* strict premonoidal functor *is a functor $F : \mathbb{C} \to \mathbb{D}$ 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*.*

Strict monoidal categories and strict premonoidal categories, along with strict functors form categories, MonCat and $\mathsf{PremonCat}$. Finally, we reach the definition of effectful categories:

▶ **Definition 20.** *A (strict)* effectful category $\mathscr{E} : \mathbb{V} \to \mathbb{C}$ *comprises*

- *a strict monoidal category $\mathbb{V}$,*
- *a strict premonoidal category $\mathbb{C}$ with the same objects as $\mathbb{V}$, $\mathbb{C}_{\mathsf{obj}} = \mathbb{V}_{\mathsf{obj}}$,*
- *an identity-on-objects strict premonoidal functor $\mathscr{E} : \mathbb{V} \to \mathbb{C}$,*
- *such that the image of $\mathscr{E} : \mathbb{V} \to \mathbb{C}$ is central.*

▶ **Definition 21.** *A* morphism of (strict) effectful categories*, $(\alpha_0, \alpha) : (\mathscr{E} : \mathbb{V} \to \mathbb{C}) \to (\mathscr{F} : \mathbb{W} \to \mathbb{D})$, comprises a strict monoidal functor $\alpha_0 : \mathbb{V} \to \mathbb{W}$ and a strict premonoidal functor $\alpha : \mathbb{C} \to \mathbb{D}$ such that $\mathscr{E} \fatsemi \alpha = \alpha_0 \fatsemi \mathscr{F}$ in $\mathsf{PremonCat}$ (eliding the inclusion of $\mathsf{MonCat} \hookrightarrow \mathsf{PremonCat}$).*

From the fact that MonCat and $\mathsf{PremonCat}$ are categories, it follows easily that:

▶ **Proposition 22.** *Effectful categories and their morphisms form a category* EffCat.

3.3 Free effectful category over an effectful signature

Our goal now is to define a functor $\mathscr{F} : \mathsf{EffGraph} \to \mathsf{EffCat}$. We will see in the next section that this functor has a right adjoint, and so constructs the *free* effectful category on an effectful signature. We begin with the free premonoidal category over a device signature.

▶ **Definition 23.** *For a device signature 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,*

$$\frac{X \in V_G^*}{1_X : X \to X}\ \textit{ID} \qquad \frac{X, Y \in V_G^* \qquad g : W \to Z \in E_G}{X \triangleright g \triangleleft Y : X \otimes W \otimes Y \to X \otimes Z \otimes Y}\ \textit{WHISK} \qquad \frac{u : P \to Q \qquad v : Q \to R}{u \fatsemi v : P \to R}\ \textit{COMP}$$

- *and $\equiv$ is the least congruence for $\fatsemi$ generated by $(u \fatsemi v) \fatsemi w = u \fatsemi (v \fatsemi w)$, $1_X \fatsemi u = u = u \fatsemi 1_Y$ whenever these are well typed, along with equations allowing (whiskerings of) orthogonal generating arrows to interchange, i.e. for every pair, $f : U \to V$ and $g : U' \to V'$, of arrows in E_G that are orthogonal, $\mathsf{dev}(f) \cap \mathsf{dev}(g) = \varnothing$, and each triple of objects X, Y, Z in V_G^*,*

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

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

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

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

It is easily verified that these mappings are well-defined, functorial, and satisfy the required coherence equations of Definition 17.

The morphisms of $\mathcal{F}G$ 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 6. 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 $\mathcal{F}G$ via the device assignment described above.

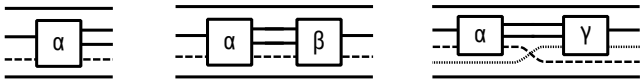


Figure 6 (Left to right): Whiskering of a generating arrow by a resource, composition of morphisms with matching devices, composition of morphisms with distinct devices. Composition gives a morphism whose devices are the union of the components.

▶ **Definition 24.** *Device assignment extends inductively from a device signature G to a well-defined function on the morphisms of $\mathcal{F}G$ (Definition 23) as follows*

$$\mathsf{dev}_{\mathcal{F}G}(f) := \begin{cases} \varnothing & \text{if } f = 1_A, \\ \mathsf{dev}_G(f') & \text{if } f = X \triangleright f' \triangleleft Y, \\ \mathsf{dev}_{\mathcal{F}G}(g) \cup \mathsf{dev}_{\mathcal{F}G}(h) & \text{if } f = g \fatsemi h. \end{cases}$$

▶ **Lemma 25.** *Let f, g be two morphisms of $\mathcal{F}G$. If f and g are orthogonal, $\mathsf{dev}_{\mathcal{F}G}(f) \cap \mathsf{dev}_{\mathcal{F}G}(g) = \varnothing$, then $f \parallel g$ and $g \parallel f$.*

Proof. See Appendix A. ◀

▶ **Example 26.** As noted in Example 9, a device signature G with empty set of objects is precisely a distribution of an alphabet. In this case, $\mathcal{F}G$ 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 signatures, $\alpha : G \to H$, induces a strict premonoidal functor, $\mathcal{F}(\alpha) : \mathcal{F}G \to \mathcal{F}H$, defined as follows. The action on objects is the lifting of α_V to lists, $\alpha_V^* : V_G^* \to V_H^*$. The action on morphisms is given by*

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

and this action is well-defined.

Proof. Well-definedness follows from the fact $\mathcal{F}G$ and $\mathcal{F}H$ 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 23 and Lemma 27 define a functor* $\mathcal{F} : \mathsf{DevGraph} \to \mathsf{PremonCat}$.

To define the free effectful category over an effectful signature, we simply combine the construction of the free monoidal category on a monoidal signature (Construction 48) with the construction of the free premonoidal category on a device signature (Definition 23), as follows.

▶ **Definition 29.** *Let $\mathcal{G} : V \to C$ be an effectful signature. This generates the following effectful category $\mathscr{F}\mathcal{G} : \mathcal{F}_\otimes V \to \mathcal{F}C$, which Proposition 30 checks is well-defined.*

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

$$\mathscr{F}\mathcal{G}(g_1 \otimes g_2) := (\mathscr{F}\mathcal{G}(g_1) \rtimes A_2)(B_1 \ltimes \mathscr{F}\mathcal{G}(g_2)) = (A_1 \ltimes \mathscr{F}\mathcal{G}(g_2))(\mathscr{F}\mathcal{G}(g_1) \rtimes B_2).$$

For this equality to hold, we must have that $\mathscr{F}\mathcal{G}(g_1)$ and $\mathscr{F}\mathcal{G}(g_2)$ 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 $\mathscr{F}\mathcal{G}(g)$ 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 signature, free monoidal category and free premonoidal category, we have that $(\mathcal{F}C)_{\mathsf{obj}} = (\mathcal{F}_\otimes V)_{\mathsf{obj}} := V^*_{\mathsf{obj}}$. We must check that $\mathscr{F}\mathcal{G} : \mathcal{F}_\otimes V \to \mathcal{F}C$ is a strict premonoidal functor with central image. $\mathscr{F}\mathcal{G}$ is identity-on-objects and so a monoid homomorphism. Furthermore it preserves left-whiskering since

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

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

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

- *the underlying strict monoidal functor is* $\mathcal{F}_\otimes\alpha_0 : \mathcal{F}_\otimes V \to \mathcal{F}_\otimes W$,
- *the underlying strict premonoidal functor* $\mathcal{F}\alpha : \mathcal{F}C \to \mathcal{F}D$ *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* $\mathscr{F} : \mathsf{EffGraph} \to \mathsf{EffCat}$.

4 Interference: from effectful categories to effectful signatures

In this section we show that every effectful category has an underlying effectful signature, 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* $\lightning\mathbb{C}$*, 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* $f \lightning g$*): that is, they do not interchange.*

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

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

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

  That is, there is an arrow $X_1, \ldots, X_n \to Y_1, \ldots, Y_m$ *in* $\mathcal{U}(\mathbb{C})$ *for every morphism* $X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$ *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* $f : X_1 \otimes ... \otimes X_n \to Y_1 \otimes ... \otimes Y_m$ *is included both in* $\mathcal{U}(\mathbb{C})(X_1 \otimes ... \otimes X_n; Y_1 \otimes ... \otimes Y_m)$ *and* $\mathcal{U}(\mathbb{C})(X_1, ..., X_n; Y_1, ..., Y_m)$*. 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,* $\mathsf{cliques}(\lightning\mathbb{C})$*, is given by the set of non-trivial maximal cliques in the interference graph,* $\lightning\mathbb{C}$*. A clique is* trivial *just when it is a singleton. A clique is maximal just in case any morphism that interferes with every other morphism of the clique is in the clique. The device assignment,* $\mathsf{dev} : E_{\mathcal{U}(\mathbb{C})} \to \mathscr{P}(\mathsf{cliques}(\lightning\mathbb{C}))$*, assigns an arrow to the subset of non-trivial maximal cliques to which it belongs.*

▶ **Lemma 35.** *For* $F : \mathbb{X} \to \mathbb{Y}$ *a strict premonoidal functor, we have that* $f \parallel g$ *implies* $F(f) \parallel F(g)$ *(interchange is preserved), and that* $F(f) \lightning F(g)$ *implies* $f \lightning g$ *(interference is reflected).*

Proof. Immediate. ◀

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

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

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

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

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

- *monoidal signature $\mathcal{U}_\otimes(\mathbb{V})$, the underlying monoidal signature of $\mathbb{V}$ (Definition 50),*
- *device signature $\mathcal{U}(\mathbb{C})$,*
- *morphism of monoidal signatures $\mathscr{U}(\mathscr{E}) : \mathcal{U}_\otimes(\mathbb{V}) \to |\mathcal{U}(\mathbb{C})|$ defined to be identity-on-objects, and $f \mapsto \mathscr{E}(f)$ on arrows.*

Proof. 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 $\mathsf{dev}(\mathscr{U}(\mathscr{E})(v)) = \varnothing$ for all arrows v in $\mathcal{U}_\otimes(\mathbb{V})$. By definition of $\mathscr{U}(\mathscr{E})$, we have $\mathscr{U}(\mathscr{E})(v) = \mathscr{E}(v)$. Also, by definition $\mathscr{E} : \mathbb{V} \to \mathbb{C}$ has central image, and so $\mathscr{E}(v)$ must constitute a trivial clique in $\lightning\, \mathbb{C}$. ◀

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

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

▶ **Proposition 40.** *The assignments of Propositions 38 and 39 extend to a functor* $\mathscr{U} : \mathsf{EffCat} \to \mathsf{EffGraph}$.

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 signature (Definition 23) is left adjoint to forming the underlying device signature of a premonoidal category (Definition 34). That is, $\mathcal{F} \dashv \mathcal{U} : \mathsf{PremonCat} \leftrightarrows \mathsf{DevGraph}$.*

▶ **Corollary 42.** *The adjunction of Theorem 41 extends to an adjunction between effectful signatures and effectful categories. That is, $\mathscr{F} \dashv \mathscr{U} : \mathsf{EffCat} \leftrightarrows \mathsf{EffGraph}$.*

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 signatures gives rise to a simple, concrete construction of this commuting tensor product.

Roughly speaking, given two free effectful categories generated by effectful signatures over the same monoidal signature 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 signatures. 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 signatures $\mathcal{G} : W \to C$ and $\mathcal{H} : W \to D$ over the same monoidal signature W, we construct their* commuting tensor product $\mathcal{G} \odot \mathcal{H} : W \to C \odot D$ *as follows. The monoidal signature of pure morphisms is simply W. The device signature $C \odot D$ 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,*

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

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

$$\begin{array}{ccccc} E_W & \xrightarrow{\mathcal{H}_E} & E_D & \xrightarrow{\mathsf{dev}_D} & \mathcal{P}(\mathcal{D}_D) \\ {\scriptstyle \mathcal{G}_E}\downarrow & & \downarrow{\scriptstyle R_E^{C,D}} & & \downarrow{\scriptstyle \mathcal{P}(\mathsf{inr})} \\ E_C & \xrightarrow[L_E^{C,D}]{} & E_C +_{E_W} E_D & & \mathcal{P}(\mathcal{D}_C + \mathcal{D}_D) \\ {\scriptstyle \mathsf{dev}_C}\downarrow & & & \searrow^{\mathsf{dev}_{C\odot D}} & \| \\ \mathcal{P}(\mathcal{D}_C) & \xrightarrow[\mathcal{P}(\mathsf{inl})]{} & \mathcal{P}(\mathcal{D}_C + \mathcal{D}_D) & = & \mathcal{P}(\mathcal{D}_{C\odot D}) \end{array}$$

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

▶ **Example 44.** A simple example is given by our printer example from Section 1, reproduced in Figure 7 (left), with the chosen monoidal signature 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 signature with itself is the effectful signature in Figure 7 (right), in which we have two distinct printers.

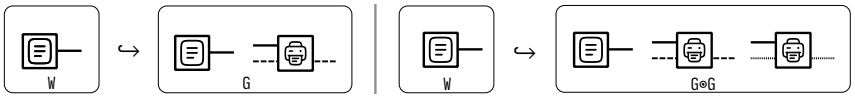


Figure 7 (Left) effectful signature corresponding to the simple "theory of a printer". (Right) Commuting tensor product of this effectful signature with itself: the theory of two printers that may be used in parallel.

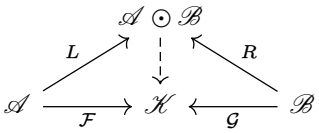
Our next goal is to show that the free effectful category over a commuting tensor product of effectful signatures is isomorphic to the commuting tensor product of the free effectful categories over those graphs. First, we recall the definition of commuting tensor product of effectful categories given 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*

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

over the same monoidal category $\mathbb{V}$, is said to be commuting *when, for every f in $\mathbb{A}$ and g in $\mathbb{B}$, we have $F(f) \parallel G(g)$, i.e. $F(f)$ and $G(g)$ interchange in $\mathbb{K}$.*

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



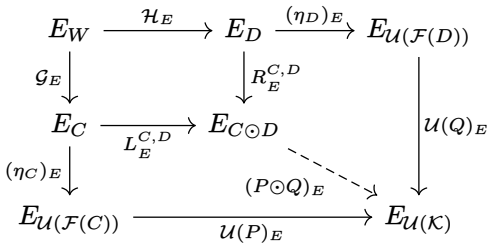
▶ **Theorem 47.** *Given effectful signatures $\mathcal{G} : W \to C$ and $\mathcal{H} : W \to D$ over the same monoidal signature W, $\mathscr{F}(\mathcal{G} \odot \mathcal{H}) \cong \mathscr{F}\mathcal{G} \odot \mathscr{F}\mathcal{H}$.*

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

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

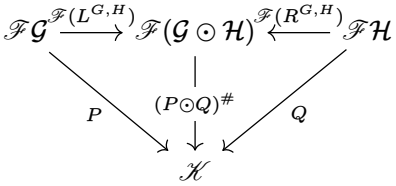
is an initial commuting cospan, where $L^{\mathcal{G},\mathcal{H}} : \mathcal{G} \to \mathcal{G} \odot \mathcal{H}$ is the morphism of effectful signatures that is identity on objects and pure generators, and given by the left pushout injection $L_E^{C,D}$ on generators (Construction 43), and similarly for $R^{\mathcal{G},\mathcal{H}} : \mathcal{H} \to \mathcal{G} \odot \mathcal{H}$. This cospan is commuting because of the way in which $\mathsf{dev}_{C \odot D}$ 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 $y \in E_D$, $L^{C,D}(x) \perp_{C \odot D} R^{C,D}(y)$. It follows that $\mathscr{F}(L^{\mathcal{G},\mathcal{H}})(f) \perp_{\mathscr{F}(\mathcal{G} \odot \mathcal{H})} \mathscr{F}(R^{\mathcal{G},\mathcal{H}})(g)$ for any morphisms f in $\mathscr{F}(\mathcal{G})$ and g in $\mathscr{F}(\mathcal{H})$, 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 $\mathscr{F}\mathcal{G} \xrightarrow{P} (\mathscr{K} : \mathcal{F}_\otimes W \to \mathcal{K}) \xleftarrow{Q} \mathscr{F}\mathcal{H}$. We obtain a function $(P \odot Q)_E : E_{C \odot D} \to E_{\mathcal{U}(\mathcal{K})}$ by the universal property of the pushout, where η denotes inclusion of generators,



This mapping on generators of the device signatures defines a morphism of effectful signatures $P \odot Q : \mathcal{G} \odot \mathcal{H} \to \mathscr{U}(\mathscr{K})$, where the fact that P and Q are a commuting cospan ensures

that $f \perp_{\mathcal{G}\odot\mathcal{H}} g \Rightarrow (P \odot Q)(f) \perp_{\mathscr{U}(\mathscr{K})} (P \odot Q)(g)$. Using the bijection of hom-sets given by Corollary 42, we obtain a morphism $(P \odot Q)^{\#} : \mathscr{F}(\mathcal{G} \odot \mathcal{H}) \to \mathscr{K}$ of EffCat such that



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

6 Conclusion and further work

We have shown how morphisms in free effectful categories can be seen as generalizations of Mazurkiewicz traces, by introducing a new notion of generating data for effectful categories, namely effectful signatures. 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 [23].

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 [22, 10], recasting our concepts from that point of view would bring our work adjacent to the literature on combining monads [12], 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 Melliès in a treatment of the local state monad [18]. 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 [23] proved that string diagrams over effectful signatures 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 signatures. 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. 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. doi:10.1016/0890-5401(87)90040-X.

4 Bob Coecke, Tobias Fritz, and Robert W Spekkens. A mathematical theory of resources. *Information and Computation*, 250:59–86, 2016. doi:10.1016/J.IC.2016.02.008.

5 Volker Diekert and Grzegorz Rozenberg. *The Book of Traces*. World Scientific, 1995. doi:10.1142/2563.

6 Matthew Earnshaw. *Languages of String Diagrams*. PhD thesis, Tallinn University of Technology, 2025. 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. 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 Philip Hackney and Marcy Robertson. On the category of props. *Applied Categorical Structures*, 23(4):543–573, August 2015. doi:10.1007/s10485-014-9369-4.

12 Martin Hyland, Gordon Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical computer science*, 357(1-3):70–99, 2006. doi:10.1016/J.TCS.2006.03.013.

13 Alan Jeffrey. Premonoidal categories and a graphical view of programs. *Preprint*, 1997.

14 André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.

15 Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Information and Computation*, 185(2):182–210, 2003. doi:10.1016/S0890-5401(03)00088-9.

16 Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. *DAIMI Report Series*, 6(78), July 1977. doi:10.7146/dpb.v6i78.7691.

17 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, 2005. Springer Berlin Heidelberg.

18 Paul-André Melliès. Local states in string diagrams. In *Rewriting and Typed Lambda Calculi: Joint International Conference*, pages 334–348. Springer, 2014. doi:10.1007/978-3-319-08918-8_23.

19 John Power. Premonoidal categories as categories with algebraic structure. *Theoretical Computer Science*, 278(1):303–321, 2002. Mathematical Foundations of Programming Semantics 1996. doi:10.1016/S0304-3975(00)00340-6.

20 John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Math. Struct. Comput. Sci.*, 7(5):453–468, 1997. doi:10.1017/S0960129597002375.

21 John Power and Hayo Thielecke. Closed freyd- and κ-categories. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming*, pages 625–634, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

22 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.

23 Mario Román and Paweł Sobociński. String diagrams for premonoidal categories. *Logical Methods in Computer Science*, Volume 21, Issue 2, April 2025. doi:10.46298/lmcs-21(2:9)2025.

24 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, pages 179–192, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2429069.2429091.

25 Wieslaw Zielonka. Notes on finite asynchronous automata. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 21(2):99–135, 1987. doi:10.1051/ITA/1987210200991.

A Omitted details

▶ **Lemma 25.** *Let f, g be two morphisms of $\mathcal{F}G$. If f and g are orthogonal, $\mathsf{dev}_{\mathcal{F}G}(f) \cap \mathsf{dev}_{\mathcal{F}G}(g) = \varnothing$, 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 $f = f_1 ⨟ f_2$, then by assumption we obtain $\mathsf{dev}_{\mathcal{F}G}(f_1) \cap \mathsf{dev}_{\mathcal{F}G}(g) = \mathsf{dev}_{\mathcal{F}G}(f_2) \cap \mathsf{dev}_{\mathcal{F}G}(g) = \varnothing$, and by hypothesis $f_1 \| g$ and $f_2 \| g$. Say $f_1 : A \to B, f_2 : B \to C, g : D \to E$. We need to show $((f_1 ⨟ f_2) \rtimes D) ⨟ (C \ltimes g) = (A \ltimes g) ⨟ ((f_1 ⨟ f_2) \rtimes E)$. By the functoriality of $\rtimes$ and associativity of $⨟$, the left-hand side is equal to $(f_1 \rtimes D) ⨟ ((f_2 \rtimes D) ⨟ (C \ltimes g))$. Using the two induction hypotheses, this becomes $(A \ltimes g) ⨟ ((f_1 \rtimes E) ⨟ (f_2 \rtimes E))$ which is finally equal to the right-hand side by the functoriality of $\rtimes$. The argument for $g \| f$ is symmetric, as well as the case when g is a composite. ◀

The following tautological construction of the free strict monoidal category on a monoidal signature is standard (see e.g. [11, Appendix A]).

▶ **Construction 48** (Free strict monoidal category)**.** *For a monoidal signature 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:*

$$\frac{f : x \to y \in G}{f : x \to y}\ \textit{GEN} \qquad \frac{A \in V_G^*}{1_A : A \to A}\ \textit{ID} \qquad \frac{f : A \to B \qquad g : B \to C}{fg : A \to C}\ \textit{COMP}$$

$$\frac{f_1 : A_1 \to B_1 \qquad f_2 : A_2 \to B_2}{f_1 \otimes f_2 : A_1 \otimes A_2 \to B_1 \otimes B_2}\ \textit{TENSOR}$$

subject to the equations,

$$(fg)h = f(gh) \qquad 1_A f = f = f 1_B \qquad (1_A \otimes 1_B) = 1_{A \otimes B}$$

$$(f_1 \otimes f_2)(g_1 \otimes g_2) = (f_1 g_1) \otimes (f_2 g_2) \qquad (f \otimes g) \otimes h = f \otimes (g \otimes h)$$

- ***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 signatures is given by the evident inductive extension.

▶ **Proposition 49.** *The construction of $\mathcal{F}_{\otimes}$ defines a functor $\mathcal{F}_{\otimes} : \mathsf{MonGraph} \to \mathsf{MonCat}$.*

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

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

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

▶ **Proposition 51.** *Definition 50 defines a functor* $\mathcal{U}_{\otimes} : \mathsf{MonCat} \to \mathsf{MonGraph}$.

▶ **Proposition 52.** *There is an adjunction* $\mathcal{F}_{\otimes} \dashv \mathcal{U}_{\otimes} : \mathsf{MonCat} \leftrightarrows \mathsf{MonGraph}$.

Proof. A left adjoint to $\mathcal{U}_{\otimes}$ is constructed via string diagrams by Joyal and Street [14]. ◀

▶ **Theorem 41.** *The free construction of premonoidal categories over a device signature (Definition 23) is left adjoint to forming the underlying device signature of a premonoidal category (Definition 34). That is,* $\mathcal{F} \dashv \mathcal{U} : \mathsf{PremonCat} \leftrightarrows \mathsf{DevGraph}$.

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

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

On arrows, recall that by definition,

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

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

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

where $\triangleright f \triangleleft$ denotes the primitive left-right whiskering of f by the empty list, as in Definition 23. It is easy to check that this is a morphism of the underlying monoidal signatures. 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 were they 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 $\alpha : G \to H$ in $\mathsf{DevGraph}$ we have:

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

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

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

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:

$$((\eta_H)_E \circ \alpha_E)f = (\eta_H)_E(\alpha_E(f)) = \mathsf{in}\ \triangleright \alpha_E(f) \triangleleft = \mathsf{in}\ F(\alpha)(\triangleright f \triangleleft)$$
$$= \mathcal{U}(F(\alpha))_E(\mathsf{in}\ \triangleright f \triangleleft) = \mathcal{U}(F(\alpha))_E(\eta_G f) = (\mathcal{U}(F(\alpha))_E \circ (\eta_G))f$$

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

We now turn our attention to the counit $\varepsilon : \mathcal{F} \circ \mathcal{U} \to 1_{\mathsf{PremonCat}}$, where $\varepsilon_{\mathbb{C}} : \mathcal{F}(\mathcal{U}(\mathbb{C})) \to \mathbb{C}$ has action on objects $(\varepsilon_{\mathbb{C}})(A_1, \ldots, A_n) := A_1 \otimes \cdots \otimes A_n$, 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:

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

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 $F : \mathbb{C} \to \mathbb{D}$ in $\mathsf{PremonCat}$ we have:

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

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

$$(F \circ \varepsilon_{\mathbb{C}})(A_1, \ldots, A_n) = F(A_1 \otimes \ldots \otimes A_n)$$
$$= F(A_1) \otimes \ldots \otimes F(A_n) = (\varepsilon_{\mathbb{D}} \circ \mathcal{F}(\mathcal{U}(F)))(A_1, \ldots, A_n).$$

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

$$(F \circ \varepsilon_{\mathbb{C}})(X \triangleright \mathsf{in} f' \triangleleft Y) = F(\varepsilon_{\mathbb{C}}(X) \triangleright f' \triangleleft \varepsilon_{\mathbb{C}}(Y)) = F(\varepsilon_{\mathbb{C}}(X)) \triangleright F(f') \triangleleft F(\varepsilon_{\mathbb{C}}(Y))$$
$$= \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(X)) \triangleright F(f') \triangleleft \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(Y))$$
$$= \varepsilon_{\mathbb{D}}(\mathcal{F}(\mathcal{U}(F))(X \triangleright \mathsf{in} f' \triangleleft Y)) = (\varepsilon_{\mathbb{D}} \circ \mathcal{F}(\mathcal{U}(F)))(X \triangleright \mathsf{in} f' \triangleleft Y).$$

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

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

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

For the first triangle, we have:

$$(\varepsilon\mathcal{U} \circ \mathcal{U}\eta)_{\mathbb{C}}(f) = \varepsilon\mathcal{U}_{\mathbb{C}}(\eta_{\mathcal{U}\mathbb{C}}(f)) = \mathcal{U}(\varepsilon_{\mathbb{C}})(\triangleright \mathsf{in} f \triangleleft) = f = 1_{\mathcal{U}(\mathbb{C})}(f),$$

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, $f = X \triangleright \mathsf{in} f' \triangleleft Y$,

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

where for $X = X_1, ..., X_n$, we denote by $[X]$ the list $[X_1], ..., [X_n]$. 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 signatures and effectful categories. That is,* $\mathscr{F} \dashv \mathscr{U} : \mathsf{EffCat} \leftrightarrows \mathsf{EffGraph}$.

Proof. We shall make use of the adjunction of Theorem 41 and the adjunction between monoidal signatures and monoidal categories ([14]). Let $\mathcal{E} : V \to G$ be an effectful signature. We first define the components of the unit, namely a morphism of effectful signatures

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

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

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

$$\varepsilon_{\mathscr{E}} : (\mathscr{F} \circ \mathscr{U})\mathscr{E} \to \mathscr{E}$$

by letting the component between the monoidal categories be given by the counit of the adjunction between monoidal signatures and monoidal categories, $\varepsilon^{\otimes}_{\mathbb{C}} : \mathcal{F}_{\otimes}\mathcal{U}_{\otimes}\mathbb{C} \to \mathbb{C}$, 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. ◀