Presentations of Premonoidal Categories by Devices
Extended Abstract

Matt Earnshaw, Chad Nester, and Mario Román

Department of Software Science, Tallinn University of Technology, Estonia

Introduction: combining process theories and effects

Monoidal categories can be understood as process theories: processes (morphisms) transform resources (objects). Given resources P and Q, we have the joint resource $P \otimes Q$, and likewise for morphisms. We use *string diagrams* [JS91] to depicit and reason about such processes (Figure 1). Strings are resources, boxes are processes, juxtaposition is tensor ($\otimes$) and joining strings is composition. Diagrams that are topologically equivalent denote equal processes.

Premonoidal categories are a refinement of monoidal categories, introduced by Power and Robinson [PR97] to model processes with effects. In premonoidal categories, *interchange* of processes does not hold globally (Figure 1): the order of effectful statements matters. Jeffrey and Román [Jef97, Rom23] showed how to use string diagrams for premonoidal categories. By threading an extra string through every effectful process, interchange by 'sliding' is prevented.

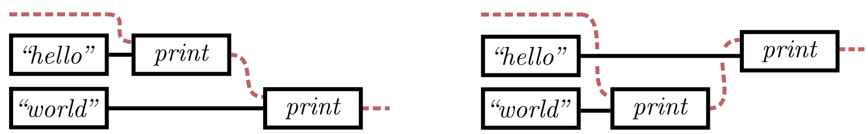

Figure 1: Dashed string threaded through *print* prevents interchange. Without it, the diagrams are equivalent, but denote different programs. With it, the diagrams are topologically distinct.

This string represents a global effect or 'runtime'. However, when modelling multiple effects, *some* effectful processes may interchange. In the presence of this global effect, equations are required to capture such interchange, limiting the reasoning we can perform by string diagram manipulation. We overcome this by extending the graphical syntax to include multiple 'effect' strings, which we term *devices*. By *device*, we mean informally: I/O peripherals, databases, memory cells, etc. In general, *devices* correspond to definite noun phrases, and resources, indefinite. If we only have one oven, *oven* $\otimes$ *oven* will be prohibited by construction. We show that any premonoidal category admits a presentation by string diagrams with multiple devices.

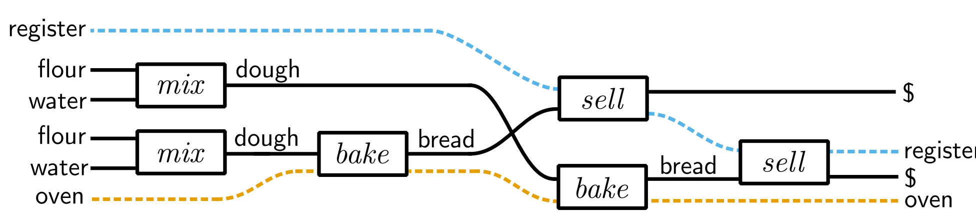

Figure 2: Flour, water, etc. are resources: we can mix in parallel. If we have only one oven and cash register, they are devices: we cannot bake in parallel, or sell in parallel. However, we can sell the first bread before or after we bake the second: these interchange.

*This research was supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19- 0001) and Estonian Research Council grant PRG1210.

Device signatures

Monoidal categories can be presented by *monoidal signatures* and equations between string diagrams over the signature [Sel10]. We augment these signatures by allowing each process to additionally use zero or more *devices*.

Definition 1. A *device signature* is given by a set of *resources* R, a set of *processes* P, a set of *devices* D, functions $s, t : P \to R^*$ assigning source and target *words* of resources to each process and a function $d : P \to \mathcal{P}(D)$ specifying a set of devices used by each process.

Graphically, we represent this data as in Figure 3. Dashed strings represent devices, and solid strings, resources. Here, we consider only one type of resource, which is left unlabelled.

Figure 3: Signature with two devices: memory cell (green) and random number generator (red).

Proposition 1. *Device signatures freely generate premonoidal categories.*

Informally, the morphisms of this category are the string diagrams in which each device appears exactly once in each vertical 'slice' through the diagram (Figures 1, 2, 4 and 5). Formally, we extend the *braid clique* and *runtime category* constructions introduced by Román [Rom23]. A *device presentation* further specifies some equations between these string diagrams. For example, we should ask that *get* ⨾ *set* equal the identity. In the next section, we show that every premonoidal category may be given such a presentation in a canonical way.

Example 1. Mazurkiewicz traces [Maz89, DR95] model the behaviour of concurrent machines. Traces generalize *words*, the behaviour of sequential machines, by allowing specified pairs of actions to commute. It follows from recent work of Earnshaw and Sobociński [ES23] that traces are morphisms in premonoidal categories over device signatures in which there are no resources (Figure 4). In traces, actions are merely names; they do not effect transformations of resources. Our setting provides the possibility of a trace theory which goes beyond merely atomic actions.

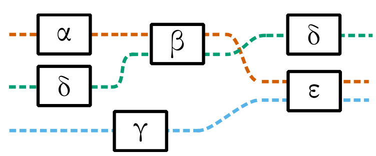

Figure 4: A trace with three devices. Actions α, δ, γ can commute, but e.g. α and β cannot. These devices may be conceived of as counters in shared memory, incremented by the actions.

Example 2. Recent work of Barrett, Heijltjes, and McCusker [BHM23, Bar23] introduces a categorical semantics for the *functional machine calculus*, a model of effectful computation based on the lambda calculus, augmented with stacks modelling different effects. This seems well-suited to presentation in terms of premonoidal categories with devices, and we adapt one of their examples. Figure 5 gives a morphism in the premonoidal category presented by the signature in Figure 3. Two random numbers are generated and sequentially stored and retrieved from the memory cell, before being added. The second random number generation can be interchanged with the first storage and retrieval, but the two random number generator calls cannot be interchanged, nor can the two uses of the memory cell.

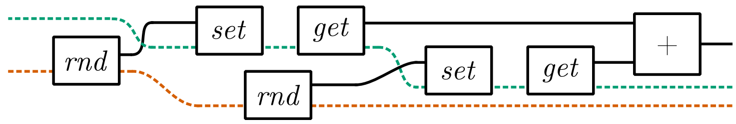

Figure 5: The second call to *rnd* may occur before or after the first *set*/*get*, but not before the first call to *rnd*. This is enforced by the device strings. Example adapted from Barrett [Bar23].

Premonoidal categories admit device presentations

Any premonoidal category can be presented by devices. The existence of a tautological presentation by one device is exactly the result of Jeffrey and Román [Jef97, Rom23]. Here we show that more refined presentations can be given, which in general can involve an infinite number of devices. We extract such a presentation via the *interference graph* of a premonoidal category.

Definition 2. The interference (or *dependence*) graph of a premonoidal category $\mathbb{C}$ has vertices the morphisms of $\mathbb{C}$, and an undirected edge (f, g) if and only if f and g do not interchange.

Roughly speaking, an edge between processes in this graph means that they cannot be executed in parallel. Each maximal clique containing more than one vertex is a device. The set of devices used by a morphism is the set of non-trivial maximal cliques to which it belongs. This defines a device signature, and we can further impose any equations holding in the category.

Proposition 2. *The interference graph of a premonoidal category $\mathbb{C}$ determines a device presentation of $\mathbb{C}$ that contains a device for each non-trivial maximal clique.*

Example 3. The *free cornering* of a monoidal category $\mathbb{A}$, models message passing between concurrent processes [Nes23]. The free cornering comprises square *cells* with resource transformations going from top to bottom, and resource passing occuring on the left and right. This gives rise to a premonoidal category $\langle\mathbb{A}\rangle$, described by Nester [Nes22]. This premonoidal category may be presented by two devices, corresponding to the left and right boundaries.

Centralizers in a premonoidal category

The *centralizer* of a subset S of a monoid M contains all elements commuting with every element in S. When $S = M$ we speak of the *centre*. In the context of premonoidal categories, the analogue of *commuting elements* is *interchanging morphisms*. The centre of a premonoidal category in this sense, is a monoidal subcategory [PR97]. We can also define *centralizers* with respect to this relation. Centralizers are now premonoidal subcategories, and every premonoidal category stratifies into a lattice of centralizers, each of which models a subset of the devices.

Proposition 3. *The centralizer of a set of morphisms in a premonoidal category is a premonoidal category.*

Proposition 4. *Every premonoidal category $\mathbb{C}$ admits a lattice of premonoidal subcategories, bounded below by its* centre*, and above by $\mathbb{C}$.*

Carette, Lemonnier, and Zamdzhiev [CLZ23] introduced *centres* for strong monads. Effectful categories, mildly generalizing premonoidal categories, are in bijection with strong promonads on (symmetric) monoidal categories [GF16, Rom23, JHH09]. Extending the centre of a monad to centralizers of strong promonads would provide a conceptual approach to centralizers. We also expect tensor products of strong promonads, which can be presented by string diagrams, to be a good setting in which to study combinations of device presentations.

References

[Bar23] Chris Barrett. *On the Simply-Typed Functional Machine Calculus: Categorical Semantics and Strong Normalisation.* PhD thesis, University of Bath, 2023. Available at https://arxiv.org/abs/2305.16073.

[BHM23] 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.

[CLZ23] Titouan Carette, Louis Lemonnier, and Vladimir Zamdzhiev. Central submonads and notions of computation: Soundness, completeness and internal languages. In *LICS*, pages 1–13, 2023.

[DR95] V Diekert and G Rozenberg. *The Book of Traces.* World Scientific, 1995.

[ES23] Matthew Earnshaw and Paweł Sobociński. String Diagrammatic Trace Theory. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GF16] Richard Garner and Ignacio López Franco. Commutativity. *Journal of Pure and Applied Algebra*, 220(5):1707–1751, 2016.

[Jef97] Alan Jeffrey. Premonoidal categories and a graphical view of programs. *Preprint*, 1997.

[JHH09] Bart Jacobs, Chris Heunen, and Ichiro Hasuo. Categorical semantics for arrows. *J. Funct. Program.*, 19(3-4):403–438, 2009.

[JS91] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991.

[Maz89] 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.

[Nes22] Chad Nester. Situated transition systems. In Kohei Kishida, editor, Proceedings of the Fourth International Conference on *Applied Category Theory,* Cambridge, United Kingdom, 12-16th July 2021, volume 372 of *Electronic Proceedings in Theoretical Computer Science*, pages 103–115. Open Publishing Association, 2022.

[Nes23] Chad Nester. Concurrent Process Histories and Resource Transducers. *Logical Methods in Computer Science*, Volume 19, Issue 1, January 2023.

[PR97] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Math. Struct. Comput. Sci.*, 7(5):453–468, 1997.

[Rom23] Mario Román. Promonads and string diagrams for effectful categories. In Jade Master and Martha Lewis, editors, Proceedings Fifth International Conference on *Applied Category Theory,* Glasgow, United Kingdom, 18-22 July 2022, volume 380 of *Electronic Proceedings in Theoretical Computer Science*, pages 344–361. Open Publishing Association, 2023.

[Sel10] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.