Rewriting with Cartesian Traced Monoidal Categories

George Kaye and Dan R. Ghica

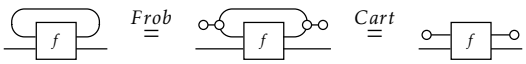
July 8, 2021

Motivation. Constructors, architects, and engineers have always enjoyed using blueprints, diagrams, and other kinds of graphical representations of their designs. These are often more than simply illustrations aiding the understanding of a formal specification: they are the specification itself. By contrast, in mathematics, diagrams have not been traditionally considered first-class citizens, although they are often used to help the reader *visualise* a construction or a proof. However, the development of new *formal* diagrammatic languages for a variety of systems such as quantum communication and computation [8], computational linguistics [9] and signal-flow graphs [2, 3], proved that diagrams can be used not just to aid understanding of proofs, but to formulate them as well. This approach has multifaceted advantages, from enabling the use of graph-theoretical techniques to aid reasoning [13] to making the teaching of algebraic concepts to younger students less intimidating [12]. *String diagrams* [25] are becoming the established mathematical language of diagrammatic reasoning, whereby equal terms are usually interpreted as isomorphic (or isotopic) diagrams. While the 'only connectivity matters' mantra is enough for reasoning about *structural* properties, properties which have computational content require a *rewriting* of the diagram. To make this possible, diagrams must be represented as combinatorial objects, such as graphs or hypergraphs, which have enough structure; the framework of *adhesive categories* [24] is often used to ensure that graph rewriting is always well-defined.

These diagrammatic languages build on an infrastructure of (usually symmetric and strict) monoidal categories [18], and in particular compact closed categories [20]. Systems modelled by morphisms in a compact closed category have a general notion of *interface port*, so that any two ports can be connected, provided the types match. This allows compact closed categories to describe systems with a flexible and refined notion of *causality*, such as quantum systems [22] or games [7]. One system that does *not* fit the compact closed structure is that of *digital circuits*, which have a stricter notion of causality: connections may only happen between ports with the same type but opposite input-output polarities. This requires a different kind of categorical setting, that of a *symmetric traced monoidal category* [15], or STMC. These categories come equipped with an explicit operation (the trace) to model causal feedback loops.

It might seem that there are ready-to-bake recipes in the literature for providing graph models to string diagrams, including for traced categories, using *string graphs* [10, 21] and *hypergraphs* [4, 26, 6]. However, the category of circuits is also *Cartesian* as it has a notion of *product*. These categories have been referred to as *traced Cartesian categories* [14] and *dataflow categories* [25, Sec.6.4] in the literature. Unfortunately, it turns out the recipes for other graph languages do not work in the presence of Cartesian product. The simple reason is that the graph models in loc. cit. are compact closed categories, and in a compact closed category the Cartesian product degenerates into a a biproduct [17], so it cannot model, for example, digital circuits which lack a biproduct. Moreover, a compact closed category with a Cartesian structure is in fact trivial [25, Sec.6.4].

The problem can be vividly seen by considering a trace constructed out of Frobenius monoids (which induce a compact closed structure), as used in the language of hypergraphs. This degenerates in the presence of Cartesian structure, as illustrated below:



To solve this problem we need to define the trace structure directly as an *atomic operation*, and prove soundness and definability for these direct definitions.

Besides the major problem above, there are some small technical issues with hypergraphs that we solve by reintroducing the concept of homeomorphism similar to that used in framed point graphs [21]. This allows us to represent the trace of the identity, which is not well-formed in vanilla hypergraphs as it is a closed loop of wires. It also means we can identify a matching of a subgraph F in a graph $\mathrm{Tr}^X(F)$ by using a monomorphism, which is essential for performing double pushout (DPO) graph rewriting.

One may ask why we cannot simply take the compact closed construction and 'rule out' those graphs that do not represent terms in an STMC. However, the act of 'ruling out' is not compositional, but rather a global check on a semantic object. This can make reasoning non-algebraic and awkward. By defining a graph structure 'from the ground up' for STMCs, we obtain a semantic domain in which we *can* reason algebraically.

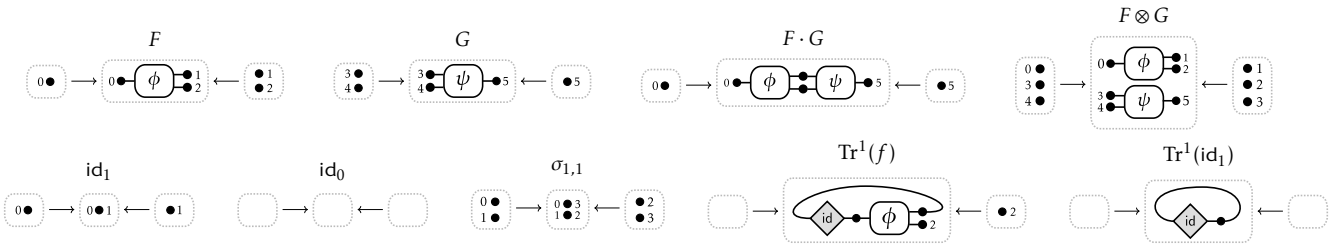


Figure 1: Operations on hypergraphs over the signature $\Sigma = \{\phi, \psi\}$.

Hypergraphs. Recent work [4, 26, 5, 6] has established *directed hypergraphs* as the graph language of string diagram rewriting, with and without a Frobenius structure. However, as we have already established the category of hypergraphs $\mathbf{Hyp}_\Sigma$ is not suitable for us. Instead we define a variant of hypergraphs known as *linear hypergraphs* with which we can define trace as an *atomic operation*. This language is *sound and complete* for STMCs: each linear hypergraph corresponds to a unique categorical term, up to the equations of the category. The use of the term *linear* stems from the fact that each vertex is the source and target of *one* edge each.

Definition 1 (Linear hypergraph)**.** *A linear hypergraph is a tuple $H = (E, S, T, \mathsf{s}, \mathsf{t}, \kappa)$ where*

- *$S, T \subset \mathbb{A}$ are finite disjoint sets of equal cardinality, of* source *and* target *vertices*
- *$E \subset \mathbb{A}$ is a finite set disjoint from S and T, of* edges.
- *s, t are subsets of S and T, each partitioned into $|E|$ (potentially empty) totally ordered parts.*
- *$\kappa : T \to S$ is a bijection between targets and sources, of* connections.

We split vertices into distinct sources and targets: this makes the concrete definitions of operations ([19, Sec. 4]) simpler. We split them into ordered disjoint parts for each edge, so each source and target vertex is associated with exactly one edge. This construction is reminiscent of the 'networks' from [16], also used for traced categories.

We label hypergraphs with generators from a monoidal signature Σ with a function $\Lambda : E \to \Sigma$, yielding *labelled* linear hypergraphs. We define the category $\mathbf{LHyp}_\Sigma$ with objects the labelled linear hypergraphs over signature Σ and morphisms their source, target, connection and label-preserving homomorphisms.

We still need to provide the *interfaces* of our hypergraphs, to order the vertices not associated with an edge. The usual way to do this is through *ordered cospans* [1]. The legs of the cospan are *discrete hypergraphs*: hypergraphs containing m source and target vertices and no edges. We write these hypergraphs simply as m: for example, $m \to F \leftarrow n$ represents a hypergraph with m inputs and n outputs. However, we cannot arbitrarily identify inputs and outputs: this would require a Frobenius structure. We adapt the condition used in [4].

Definition 2. *For $m, n \in \mathbb{N}$ and (labelled) linear hypergraph F, we say that a cospan of monomorphisms $m \xrightarrow{p} F \xleftarrow{q} n$ is* monogamous *if for all $t \in T_m$, $p_T(t) \notin \mathsf{t}_F$, and for all $s \in S_n$, $q_S(s) \notin \mathsf{s}_F$.*

Let $MCsp_D(\mathbf{LHyp}_\Sigma)$ be the cospan bicategory [4] containing only the monogamous discrete cospans over $\mathbf{LHyp}_\Sigma$.

Operations. We can now define operations on linear hypergraphs. For composition, tensor and symmetry the interpretations are obvious enough: typical instances are illustrated in Figure 1. The only subtlety is the trace: while one may be tempted to simply 'join up' the outputs and the inputs, this can lead to problems when considering the trace of the identity $\mathrm{Tr}^x(\mathsf{id}_x)$. This is a closed loop of wires [14]: something we cannot represent as a well-formed linear hypergraph. To solve this issue, we reintroduce the notion of homeomorphism from [21], to introduce *identity edges*. We can introduce an identity edge at any vertex with an *expansion* and remove them with a *smoothing*, unless this would form a closed loop. We call a hypergraph with no possible smoothings *minimal*. This means we can the trace of the identity as an edge with connected source and target, as in Figure 1.

Soundness and completeness. We propose linear hypergraphs as a graphical language for STMCs. We focus on traced PROPs [23], categories with natural numbers as objects and addition as tensor product: from now on we fix a traced PROP $\mathbf{Term}_\Sigma$, freely generated over signature Σ. The first step is to translate from terms into hypergraphs.

Definition 3 (Interpretation functor)**.** *We define the interpretation functor from terms to labelled linear hypergraphs as the identity-on-objects traced monoidal functor $[\![-]\!] : \boldsymbol{Term}_\Sigma \to MCsp_D(\boldsymbol{LHyp}_\Sigma)$:*

- *For a generator $\phi : m \to n \in \Sigma$, $[\![\phi]\!] = m \to F \leftarrow n$, where F is the linear hypergraph with one edge ϕ.*
- *$f \cdot g$, id_n, $f \otimes g$, $\sigma_{m,n}$ and $Tr^x(f)$ are defined as their corresponding operations on linear hypergraphs .*

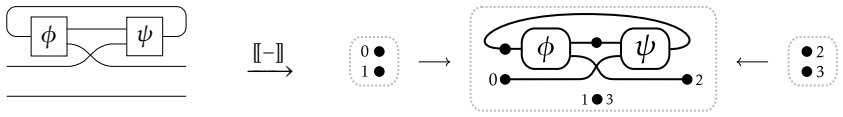


Figure 2: Interpreting a categorical term as a cospan of linear hypergraphs

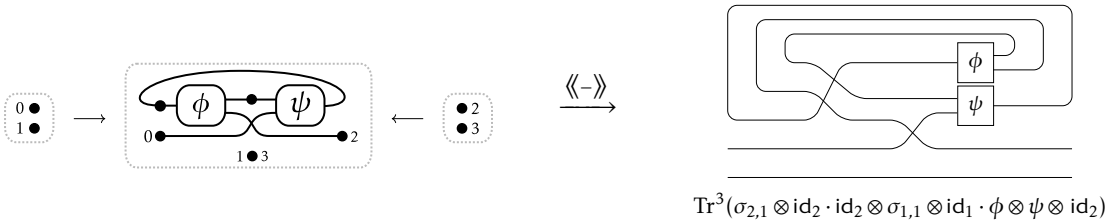


Figure 3: A hypergraph and its corresponding categorical term.

The interpretation functor is illustrated in Figure 2. For us to have soundness we must show that all axioms of STMCs (see e.g. [15, Sec. 3]) hold in the language of linear hypergraphs.

Theorem 4 (Soundness)**.** *For any morphisms $f, g \in \boldsymbol{Term}_\Sigma$, if $f = g$ under the equational theory of the category then their interpretations as cospans of labelled linear hypergraphs are isomorphic, $[\![f]\!] \equiv [\![g]\!]$.*

The converse is *completeness*: every valid linear hypergraph must correspond to a unique term in $\mathbf{Term}_\Sigma$. First we show *definability*: there exists a functor from which we can recover at least one categorical term from a cospan of linear hypergraphs. We fix an edge order $\leq$ on the edges and use this to define a tensor stack of the corresponding generators. Then we trace around the outputs of these generators, and define a shuffle of symmetries and identities to connect them to the appropriate inputs. The procedure is demonstrated in Figure 3.

Definition 5 (Definability functor)**.** *We define the definability functor as the identity-on-objects traced monoidal functor $\langle\!\langle - \rangle\!\rangle : MCsp_D(\boldsymbol{LHyp}_\Sigma) \to \boldsymbol{Term}_\Sigma$ with its action defined for a given edge order $\leq$ as*

$$\langle\!\langle m \to F \leftarrow n \rangle\!\rangle_\leq = Tr^{x-m}(\mathsf{shuffle}(F)_\leq \cdot \mathsf{stack}(F)_\leq \otimes \mathsf{id}_n)$$

Proposition 6 (Definability)**.** *For any $F \in \boldsymbol{LHyp}_\Sigma$ and edge order $\leq$, then $m \to F \leftarrow n \equiv [\![\langle\!\langle m \to F \leftarrow n \rangle\!\rangle_\leq]\!]$.*

We must also show *coherence*: the resulting categorical term is the same regardless of the initial choice of edge order.

Proposition 7 (Coherence)**.** *For all orderings of edges $\leq_x$ on some $F \in \boldsymbol{LHyp}_\Sigma$,*

$$\langle\!\langle m \to F \leftarrow n \rangle\!\rangle_{\leq_1} = \langle\!\langle m \to F \leftarrow n \rangle\!\rangle_{\leq_2} = \cdots = \langle\!\langle m \to F \leftarrow n \rangle\!\rangle_{\leq_x}$$

Theorem 8 (Completeness)**.** *For any cospan of linear hypergraphs $m \to F \leftarrow n \in MCsp_D(\boldsymbol{LHyp}_\Sigma)$ there exists a unique morphism $f \in \boldsymbol{Term}_\Sigma$, up to the equations of the STMC, such that $[\![f]\!] = F$. Moreover, for any $f \in \boldsymbol{Term}_\Sigma$, $\langle\!\langle [\![f]\!] \rangle\!\rangle = f$.*

Graph rewriting. A popular approach to graph rewriting is double pushout (DPO) rewriting [11]: we use an extension of the traditional definition known as DPO rewriting *with interfaces* [5]. To ensure that rewriting is always well-defined, the framework of *adhesive categories* is often used [24]. The key property that holds in adhesive categories is that pushout complements are unique if they exist for a rewrite rule $L \overset{p}{\leftarrow} K \to R$ and matching $L \to G$, providing p is mono. Since our category of linear hypergraphs is a full subcategory of the adhesive category of 'regular' hypergraphs $\mathbf{Hyp}_\Sigma$, we can inherit some of the adhesivity.

Proposition 9. *$\boldsymbol{LHyp}_\Sigma$ is a partial adhesive category [21].*

This means we can rewrite in $\mathbf{LHyp}_\Sigma$ just as we would in $\mathbf{Hyp}_\Sigma$: we refer the reader to [4, Sec. 4] for concrete details. An example is illustrated in Figure 5.

Moreover, since we are considering hypergraphs up to homeomorphism, we must consider *matchings modulo homeomorphism*. Essentially, this means that we disregard all identity edges and only work with the minimal versions of any graphs involved. However, sometimes identity edges are essential for well-definedness of rewriting. We will now examine these cases.

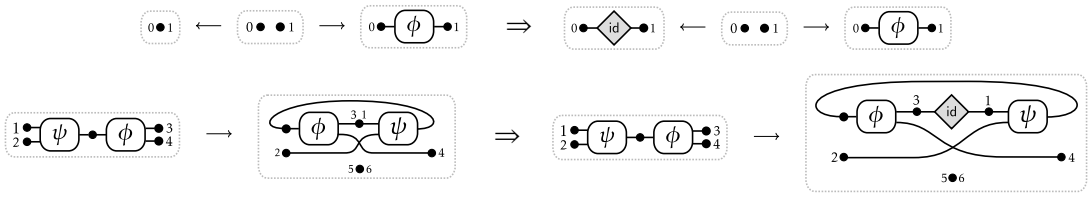


Figure 4: The two expansion rules: rule expansion and matching expansion.

We have previously established that the pushout complement is only unique for rules $L \xleftarrow{p} K \to R$ where p is mono (*left-linear* rules). Some useful rules, such as introducing an edge to an identity wire, are not left-linear, We can perform a *rule expansion* (Fig. 4) to transform it into an appropriate form.

In *partial* adhesive categories, we also require the matching $L \xrightarrow{q} G$ to be mono. When we introduce the trace it can lead to situations where the outputs can be 'bent around' to connect to the inputs, as illustrated in the top of Fig. 4. This means that m is *not* mono,so the pushout complement would not be unique; one of the results would also discard the trace completely. We perform a *matching expansion* (Fig. 4) to obtain a monomorphism $\tilde{m}$.

Definition 10. *For a rewrite span $L \xleftarrow{p} K \to R$ and morphism $L \xrightarrow{q} G \in \boldsymbol{LHyp}_\Sigma$, the homeomorphism rewriting system $\mathcal{H}$ is as follows. First, repeatedly apply smoothings until L and G are minimal. If p is not mono, expand each $(t,s) \in L$ mapped to by multiple vertices in K, and for each $(t',s') \in G$ mapped to by (t,s). If q is not mono, expand each $(t,s) \in G$ mapped to by multiple vertices in L. We write $\tilde{L}$, $\tilde{G}$, $\tilde{p}$ and $\tilde{q}$ for the resulting hypergraphs and morphisms from applying $\mathcal{H}$.*

Proposition 11. *$\mathcal{H}$ is terminating.*

Proof. Performing smoothings always reduces the size of the graph, and $\tilde{p}$ and $\tilde{q}$ are always mono. □

To consider if a hypergraph G is a candidate for rewriting with rule $L \leftarrow K \to R$, we first rewrite with $\mathcal{H}$. If there then exists a monomorphism $\tilde{L} \to \tilde{G}$, we say there is a *mono modulo homeomorphism* $L \to G$. If this mono has a pushout complement, we call it a *matching modulo homeomorphism*.

Theorem 12 (Rewriting)**.** *For $m \to G \leftarrow n \in MCsp_D(\boldsymbol{LHyp}_\Sigma)$, span $L \xleftarrow{p} m+n \to R \in \boldsymbol{LHyp}_\Sigma$ with p mono, and matching monomorphism $L \to G \in \boldsymbol{LHyp}_\Sigma$, the rewriting procedure yields a unique cospan $m \to H \leftarrow n \in MCsp_D(\boldsymbol{LHyp}_\Sigma)$.*

A benefit of rewriting with *linear* hypergraphs is that *every* monomorphism in $\mathbf{LHyp}_\Sigma$ can act as a matching in the rewriting procedure, as the oft-required 'no-dangling-hyperedges' condition always holds in our context.

Theorem 13. *All monomorphisms modulo homeomorphism in $\boldsymbol{LHyp}_\Sigma$ are matchings modulo homeomorphism.*

We can then conclude the final rewriting result:

Theorem 14. *For a set of axioms $\mathcal{E}$ in $\boldsymbol{Term}_\Sigma$ and $g,h \in \boldsymbol{Term}_\Sigma$, $g = h$ by the laws of STMCs and $\mathcal{E}$ if and only if $[\![g]\!]$ rewrites to $[\![h]\!]$ using the DPO procedure.*

We can therefore reason graphically in systems modelled as morphisms in an STMC with a Cartesian structure, such as digital circuits. This will allow us to make the proofs in [13] formal, which is the next step in our work.

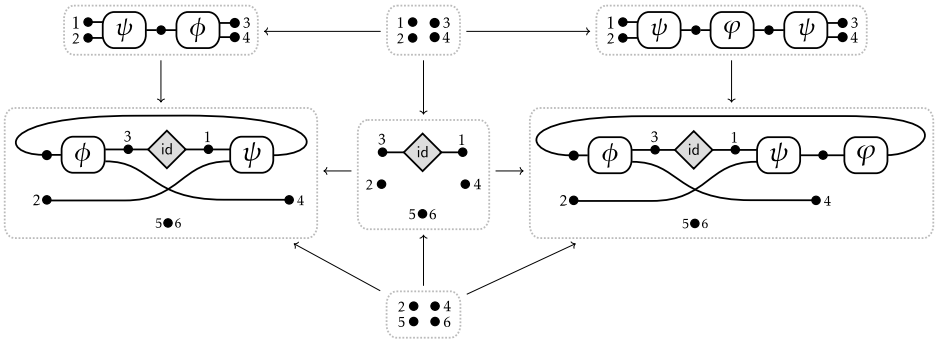


Figure 5: Applying the DPO procedure to the matching in Figure 4

References

[1] J. C. Baez and K. Courser. Structured cospans, 2020. URL https://arxiv.org/abs/1911.04630.

[2] F. Bonchi, P. Sobociński, and F. Zanasi. A categorical semantics of signal flow graphs. In *International Conference on Concurrency Theory*, pages 435–450. Springer, 2014. doi:10.1007/978-3-662-44584-6_30.

[3] F. Bonchi, P. Sobocinski, and F. Zanasi. Full abstraction for signal flow graphs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, page 515–526. Association for Computing Machinery, 2015. doi:10.1145/2676726.2676993.

[4] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. Rewriting modulo symmetric monoidal structure. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. IEEE, 2016. doi:10.1145/2933575.2935316.

[5] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi. Confluence of graph rewriting with interfaces. In *European Symposium on Programming*, pages 141–169. Springer, 2017. doi:10.1007/978-3-662-54434-1_6.

[6] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobocinski, and F. Zanasi. Rewriting with Frobenius. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 165–174, 2018. doi:10.1145/3209108.3209137.

[7] S. Castellan and P. Clairambault. Causality vs. interleavings in concurrent game semantics. In *The 27th International Conference on Concurrency Theory (CONCUR 2016)*, volume 32, pages 1 – 3214, Québec City, Canada, Aug. 2016. doi:10.4230/LIPIcs.CONCUR.2016.32.

[8] B. Coecke and A. Kissinger. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pages 28–31. Springer, 2018. doi:10.1007/978-3-319-91376-6_6.

[9] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning, 2010. URL https://arxiv.org/abs/1003.4394.

[10] L. Dixon and A. Kissinger. Open graphs and monoidal theories. *Mathematical Structures in Computer Science*, 23:308–359, 2013. doi:10.1017/S0960129512000138.

[11] H. Ehrig, M. Pfender, and H. J. Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 167–180. IEEE, 1973. doi:10.1109/SWAT.1973.11.

[12] D. R. Ghica. A knot theory for eight year olds. *Mathematical Teaching*, (264-268), 2018. URL https://www.atm.org.uk/Mathematics-Teaching-Journal-Archive/149275.

[13] D. R. Ghica, A. Jung, and A. Lopez. Diagrammatic Semantics for Digital Circuits. In *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82, pages 24:1–24:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.CSL.2017.24.

[14] M. Hasegawa. Recursion from cyclic sharing: traced monoidal categories and models of cyclic lambda calculi. In *International Conference on Typed Lambda Calculi and Applications*, pages 196–213. Springer, 1997. doi:10.1007/3-540-62688-3_37.

[15] M. Hasegawa. On traced monoidal closed categories. *Mathematical Structures in Computer Science*, 19(2):217–244, 2009. doi:10.1017/S0960129508007184.

[16] M. Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. Springer Science & Business Media, 2012. doi:10.1007/978-1-4471-0865-8.

[17] R. Houston. Finite products are biproducts in a compact closed category. *Journal of Pure and Applied Algebra*, 212(2):394–400, 2008. doi:10.1016/j.jpaa.2007.05.021.

[18] A. Joyal and R. Street. The geometry of tensor calculus, I. *Advances in mathematics*, 88(1):55–112, 1991. doi:10.1016/0001-8708(91)90003-P.

[19] G. Kaye. Rewriting graphically with symmetric traced monoidal categories, 2021. URL https://arxiv.org/abs/2010.06319.

[20] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of pure and applied algebra*, 19:193–213, 1980. doi:10.1016/0022-4049(80)90101-2.

[21] A. Kissinger. Pictures of processes: Automated graph rewriting for monoidal categories and applications to quantum computing, 2012. URL https://arxiv.org/abs/1203.0202.

[22] A. Kissinger and S. Uijlen. A categorical semantics for causal structure. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005095.

[23] S. Lack. Composing PROPs. *Theory and Applications of Categories*, 13(9):147–163, 2004. URL http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html.

[24] S. Lack and P. Sobociński. Adhesive categories. In *International Conference on Foundations of Software Science and Computation Structures*, pages 273–288. Springer, 2004. doi:10.1007/978-3-540-24727-2_20.

[25] P. Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010. doi:10.1007/978-3-642-12821-9_4.

[26] F. Zanasi. Rewriting in free hypergraph categories. *Electronic Proceedings in Theoretical Computer Science*, 263: 16–30, Dec 2017. ISSN 2075-2180. doi:10.4204/eptcs.263.2.