Rewriting with Cartesian Traced Monoidal Categories

Dan R. Ghica and **George Kaye**

May 13, 2021

Motivation. *String diagrams* [21] are becoming the established mathematical language of diagrammatic reasoning, whereby equal terms are usually interpreted as isomorphic (or isotopic) diagrams. These graphical languages build on a mathematical infrastructure of symmetric monoidal categories [15]: we can reason about compositional systems by interpreting them as morphisms in a SMC. However, while the 'only connectivity matters' mantra of string diagrams is enough for reasoning about *structural* properties, properties which have computational content in the form of additional axioms require a *rewriting* of the diagram. To make this possible, diagrams must be represented as combinatorial objects with enough structure, such as graphs or hypergraphs. We are particularly interested in the framework of *adhesive categories*, which guarantees the well-definedness of graph rewriting [20]. Our main motivation is to fully formalise prior work on diagrammatic reasoning for digital circuits [9, 10], for which we need a string diagram language of along with adhesive categorical infrastructure for rewriting.

It might seem that this is a solved problem, as sound and complete combinatorial languages for graph rewriting have already been studied as *open graphs* [7, 17] and *hypergraphs* [2, 22, 4]. However, these languages are often rooted in a *compact closed* setting, where wires can be bidirectional. This means that they can be used to describe systems with a flexible and refined notion of *causality*, such as quantum systems [18] or games [5]. In contrast, systems such as digital circuits have a stricter notion of causality, enforcing that 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* [12], or STMC. These categories come equipped with an explicit construct (the trace) to model causal feedback loops. In particular, to reason about digital circuits we require the framework of *dataflow categories*, which are STMCs in which the monoidal tensor is a Cartesian product [6, 11].

One may note that we can construct the trace by using the compact closed structure: indeed, this is the strategy used by [17, Thm. 5.4.9] for traced categories. However, this raises for us insurmountable technical problems. In general it is well known that in compact closed categories finite products automatically become *biproducts* [14]. This is enough to compromise the construction as a setting for modelling digital circuits, which do not physically satisfy the equational properties of a biproduct. But the problem runs deeper, as the Frobenius structure itself is not compatible with Cartesian product, as seen in the diagram below:

On the left, the Frobenius structure equates the splitting and joining of the wires with a feedback loop, implementing a trace structure. On the right, the splitting of the wires copies the co-unit of the Frobenius co-monoid, resulting in a degenerate circuit. To solve this problem we need to define the trace structure directly, 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 [17]. 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 also ask is why we cannot simply take the compact closed construction and 'rule out' those graphs that do not represent terms in an STMC, as with the 'positive graphs' in [17]. However, the act of 'ruling out' is not compositional but rather a global check on a semantic object, which can make reasoning non-algebraic and awkward. By defining a graph structure 'built from the ground up' for STMCs, we obtain a semantic domain in which we *can* reason algebraically.

Hypergraphs. Recent work [2, 22, 3, 4] has established *hypergraphs* as the language of string diagram rewriting, with and without a Frobenius structure. However, for the reasons elucidated above, the category of hypergraphs $\mathbf{Hyp}_\Sigma$ is not entirely suitable for us. We define a variant of hypergraphs known as *linear hypergraphs* that is *sound and complete* for symmetric traced monoidal categories: each linear hypergraph corresponds to a unique categorical term, up to the equations of the category.

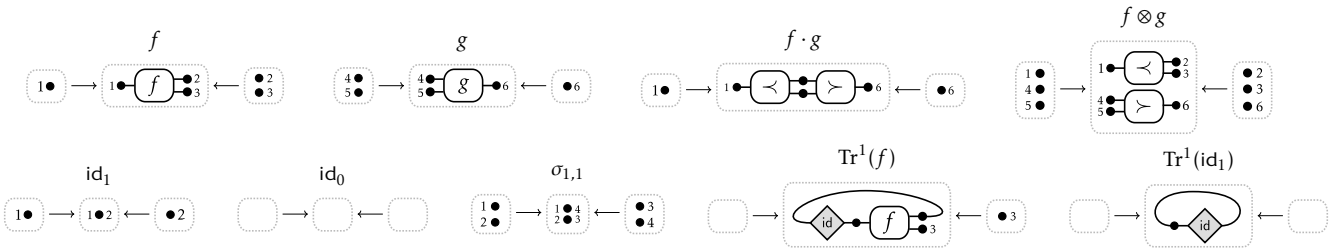

Figure 1: Operations on hypergraphs over the signature $\Sigma = \{f, g\}$.

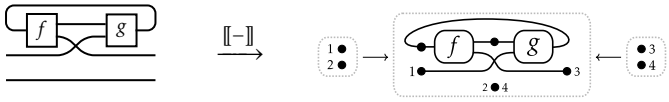

Figure 2: Interpreting a categorical term as a cospan of linear hypergraphs

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

- S, T *are finite sets of equal cardinality, of* source *and* target *vertices*
- E *is a finite set of* edges
- s, t *are partitions on subsets* $S' \subseteq S$ *and* $T' \subseteq T$ *into* $|E|$ *(potentially empty) totally ordered parts.*
- $\kappa : T \to S$ *is a* connections *bijection between targets and sources.*

We split vertices into distinct sources and targets: this makes the concrete definitions of operations ([16, Sec. 4]) simpler. These sets are split into 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 [13], 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 [2].

Definition 2. *For* $m, n \in \mathbb{N}$ *and (labelled) linear hypergraph* F*, we say that a cospan* $m \xrightarrow{p} F \xleftarrow{q} n$ *is* monogamous *if*

$$\forall s \in S_m.\, p_S(s) \in S'_F \qquad \forall t \in T_m.\, p_T(t) \notin T'_F \qquad \forall s \in S_n.\, q_S(s) \notin S'_F \qquad \forall t \in T_n.\, q_T(t) \in T'_F$$

We write $Csp(\mathbf{LHyp}_\Sigma)$ for the cospan bicategory [2] 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 [11]: something we cannot represent as a well-formed linear hypergraph. To solve this issue, we reintroduce the notion of homeomorphism from [17], to introduce *identity edges*. 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 hypergraphs as a graphical language for STMCs. We will focus on traced PROPs [19], 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* $[\![-]\!] : \boldsymbol{Term}_\Sigma \to Csp(\boldsymbol{LHyp}_\Sigma)$*:*

- *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 .*

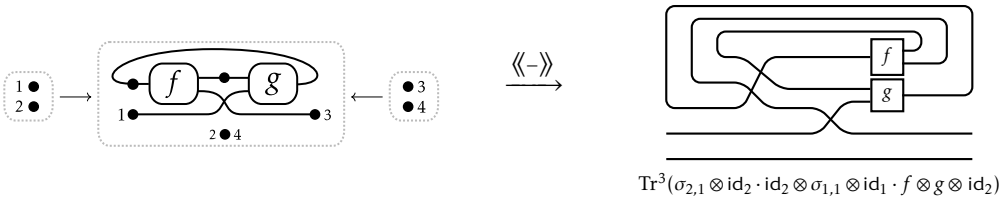

Figure 3: A hypergraph and its corresponding categorical term.

The interpretation functor is illustrated in Figure 2. For us to have soundness we must show that all axioms of STMCs (see e.g. [12, 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 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 $\langle\!\langle - \rangle\!\rangle : Csp(\boldsymbol{LHyp}_\Sigma) \to \boldsymbol{Term}_\Sigma$ 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 $m \to F \leftarrow n \equiv [\![\langle\!\langle m \to F \leftarrow n \rangle\!\rangle_\leq]\!]$.*

To conclude completeness, we must also show *coherence*: that the resulting categorical term is the same regardless of the initial choice of edge order $\leq$.

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 $m \to F \leftarrow n \in Csp(\boldsymbol{LHyp}_\Sigma)$ 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 [8]: we use an extension of the traditional definition known as DPO rewriting *with interfaces* [3]. To ensure that rewriting is always well-defined, the framework of *adhesive categories* is often used [20]. 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 [17].*

This means we can rewrite in $\mathbf{LHyp}_\Sigma$ just as we would in $\mathbf{Hyp}_\Sigma$: we refer the reader to [2, Sec. 4] for concrete details. An example is illustrated in Figure 6.

In (partial) adhesive categories, rewriting is only well-defined for rewrite spans where the left leg is mono (*left-linear* spans): otherwise the pushout complement is not unique. However, these spans are occasionally desirable, such as when rewriting the identity. In [2] the use of such spans is permitted due to the Frobenius structure, and in the symmetric monoidal case this problem is eliminated by use of *boundary complements*. We take an alternative approach by using our already introduced notion of homeomorphism: we can always rewrite a problematic span to a left-linear span by adding an identity edge, as shown in Figure 4.

Moreover, in partial adhesive categories the matching from the left hand side of the rewrite rule L to the larger graph G must also be a monomorphism. However, our traced structure can lead to situations where $L \to G$ is not a monomorphism. For example, a rule can be 'bent around' such that its output connects to its inputs, as shown in Figure 5. Once again we can solve this by using homeomorphism to obtain the required monomorphism.

Theorem 10 (Rewriting). *For $m \to G \leftarrow n \in Csp(\boldsymbol{LHyp}_\Sigma)$, 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 $m \to H \leftarrow n \in Csp(\boldsymbol{LHyp}_\Sigma)$.*

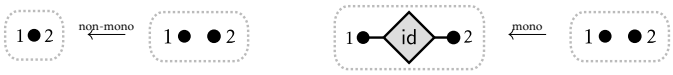

Figure 4: Using a homeomorphism to create a left-linear rewrite rule

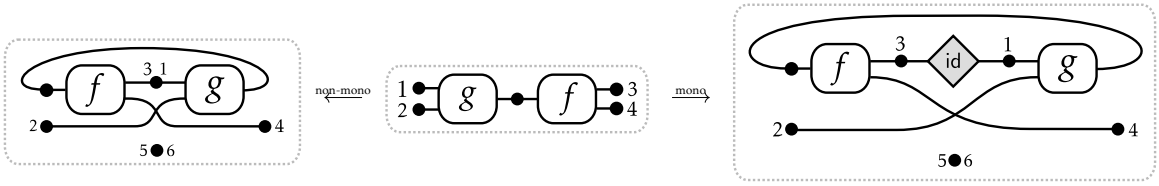

Figure 5: Using a homeomorphism to identify a matching monomorphism

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 11 (Matchings)**.** *All monomorphisms in* $\boldsymbol{LHyp}_\Sigma$ *are matchings.*

We can then conclude the final rewriting result:

Theorem 12. *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, such as digital circuits. This will allow us to make the proofs in [10] formal, which is the next step in our work.

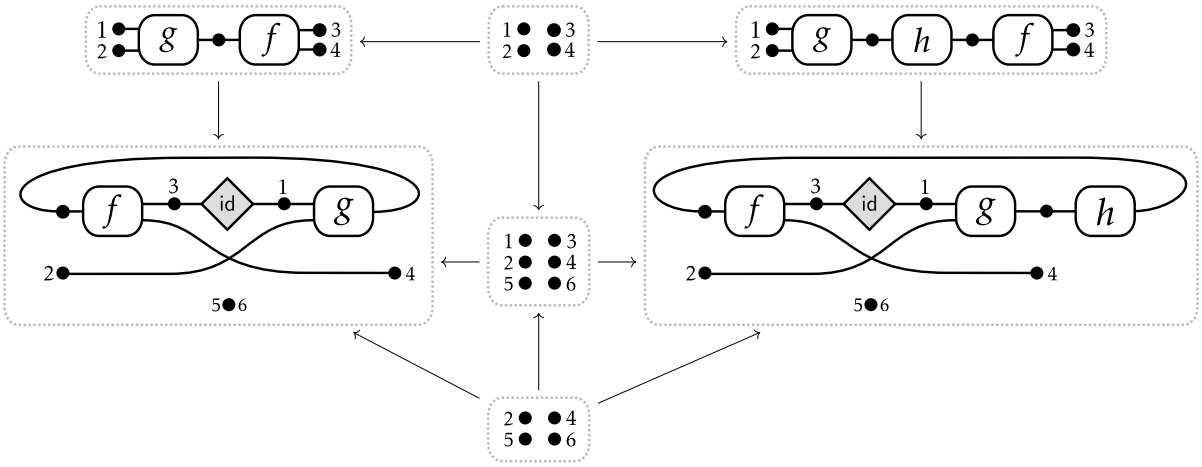

Figure 6: Applying the DPO procedure to the matching in Figure 5

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] V. E. Cazanescu and G. Stefanescu. Feedback, iteration, and repetition. In *Mathematical Aspects of Natural and Formal Languages*, 1994. doi:10.1142/9789814447133_0003.

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

[8] 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.

[9] D. R. Ghica and A. Jung. Categorical semantics of digital circuits. In *Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design*, pages 41–48. FMCAD Inc, 2016. doi:10.1109/FMCAD.2016.7886659.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.