Data-Parallel Algorithms for String Diagrams

Paul Wilson[1] and Fabio Zanasi[2]

[1]Independent

[2]University College London and University of Bologna

We give parallel algorithms for string diagrams represented as *structured cospans of ACSets*. Specifically, we give *linear* (sequential) and *logarithmic* (parallel) time algorithms for composition, tensor product, construction of diagrams from arbitrary Σ-terms, and application of functors to diagrams.

Our datastructure can represent morphisms of both the free symmetric monoidal category over an arbitrary signature as well as those with a chosen Special Frobenius structure. We show how this additional (hypergraph) structure can be used to map diagrams to *diagrams of optics*. This leads to a case study in which we define an algorithm for efficiently computing symbolic representations of gradient-based learners based on reverse derivatives.

The work we present here is intended to be useful as a *general purpose* datastructure. Implementation requires only *integer arrays* and well-known algorithms, and is data-parallel by constuction. We therefore expect it to be applicable to a wide variety of settings, including embedded and parallel hardware and low-level languages.

1 Introduction

String diagrams are a formal graphical syntax [13] for representing morphisms of monoidal categories which is now widely used (see for example [7, 8, 9, 2]). The purpose of this paper is to make string diagrams not just a convenient notation for algebraic reasoning, but also an efficient general-purpose tool in computing with graphical structures in a compositional manner. To that end, the datastructures and algorithms we define satisfy the following desiderata.

Fast and data-parallel. Our algorithms are data-parallel by construction, and have *linear* (sequential) and *logarithmic* (parallel) time complexities.

Minimal primitives. Our datastructures are defined in terms of simple integer arrays. Moreover, we assume only a small number of simple, well-known primitive operations (e.g., prefix sum). This makes it possible to implement our algorithms in a wide variety of settings, such as embedded and parallel (i.e., GPU) hardware.

Simple to implement correctly. Key parts of our datastructure are defined in terms of the recent construction of *ACSets* [11]. Consequently, implementations are essentially the same as their categorical definitions, making it easier to ensure correctness.

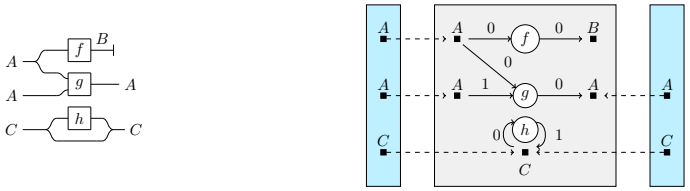
A number of representations of string diagrams such as the wiring diagrams of Catlab [10] have been explored in the literature. Our goals most closely align with the 'hypergraph adjacency representations' of [15]: we aim to make string diagrams useful as a general purpose 'scalable combinatorial syntax'. For example, we hope that our implementation serves as an alternative in cases where a programmer would currently use a tree or directed graph.

However, the primary motivating application for our work is in representing gradient-based learners as optics, as described in [5]. In particular, this motivates perhaps the most significant extension to [15]: our datastructures can 'natively' represent morphisms of the free symmetric monoidal category over a signature with a chosen Special Frobenius monoid.

We now give a brief overview of the main contributions of our paper [16].

1.1 String Diagrams as Structured Cospans of ACSets

Our datastructure represents morphisms of $\mathsf{Free}_{\Sigma+\mathbf{Frob}}$ (the free symmetric monoidal category over the signature $\Sigma+\mathbf{Frob}$) as structured cospans [1] of ACSets [11]. Diagrams represented in this way form a symmetric monoidal category $\mathsf{Diagram}_{\Sigma+\mathbf{Frob}}$ which is isomorphic to $\mathsf{Free}_{\Sigma+\mathbf{Frob}}$. Pictured below is a string diagram in $\mathsf{Free}_{\Sigma+\mathbf{Frob}}$ (left) and its representation as a structured cospan (right).

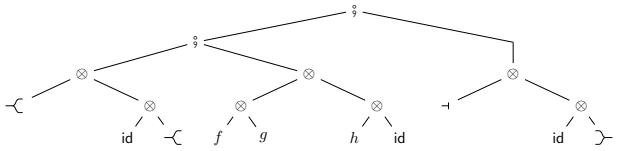


(1)

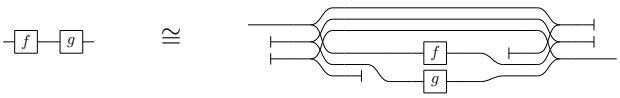
The center box, in grey, depicts a bipartite multigraph modelling the 'internal wiring' of the string diagram, where ■-nodes represent the *wires* of the diagram, and ∘-nodes the *operations*.

1.2 Data-Parallel Algorithms

We give data-parallel algorithms with *linear* (sequential) and *logarithmic* (parallel) time complexity for composition, tensor product, functor application, and for constructing a diagram from a Σ-term. Note that the latter case is required to efficiently construct a diagram from a Σ-term such as the one below, which corresponds to the diagram in (1).



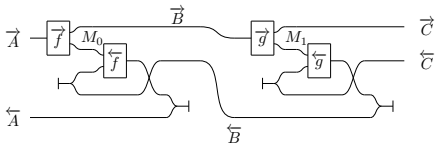
The algorithm given for applying functors to diagrams exploits the hypergraph structure of $\mathsf{Diagram}_{\Sigma+\mathbf{Frob}}$. We define 'Frobenius decompositions' for this purpose (see below) which essentially separate the wires and operations of a diagram.



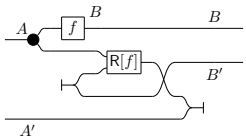
Applying a functor to diagrams of this kind then involves only its action on spiders and operations.

1.3 Diagrams of Optics and Reverse Derivatives

The hypergraph structure of $\mathsf{Diagram}_{\Sigma+\mathbf{Frob}}$ can also be used to efficiently compute composition of *optics* [6, 12], as below.



This allows modeling applications with 'bidirectional information flow' such as [5, 14, 3]. In our specific example, it allows us to define to an efficient algorithm for taking reverse derivatives [4] (see below) and modeling gradient-based learners in general.



Using our algorithm for applying functors to diagrams, this allows us to map diagrams to *diagrams of optics*, and consequently gives an algorithm for taking reverse derivatives of diagrams in linear (sequential) and logarithmic (parallel) time. More generally, this allows for mapping diagrams with *feedback* in terms of the hypergraph structure to *optics with feedback*. That is, systems with notions of both bidirectional information flow and recursion or iteration.

References

[1] John C. Baez and Kenny Courser. Structured cospans. 2019. DOI: 10.48550/ARXIV.1911.04630. URL https://arxiv.org/abs/1911.04630.

[2] Filippo Bonchi, Pawel Sobociński, and Fabio Zanasi. A Survey of Compositional Signal Flow Theory. In *Advancing Research in Information and Communication Technology*, volume AICT-600, pages 29–56. 2021. DOI: 10.1007/978-3-030-81701-5_2. URL https://hal.inria.fr/hal-03325995. part : TC 1: Foundations of Computer Science.

[3] Matteo Capucci, Neil Ghani, Jé rémy Ledent, and Fredrik Nordvall Forsberg. Translating extensive form games to open games with agency. *Electronic Proceedings in Theoretical Computer Science*, 372:221–234, nov 2022. DOI: 10.4204/eptcs.372.16. URL https://doi.org/10.4204%2Feptcs.372.16.

[4] Robin Cockett, Geoffrey Cruttwell, Jonathan Gallagher, Jean-Simon Pacaud Lemay, Benjamin MacAdam, Gordon Plotkin, and Dorette Pronk. Reverse derivative categories, 2019.

[5] G. S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning, 2021. URL https://arxiv.org/abs/2103.01931.

[6] Bruno Gavranović. Space-time tradeoffs of lenses and optics via higher category theory, 2022.

[7] Fabrizio Genovese and Jelle Herold. A categorical semantics for hierarchical petri nets, 2021.

[8] Fabrizio Romano Genovese, Fosco Loregian, and Daniele Palombi. Nets with mana: A framework for chemical reaction modelling, 2021.

[9] Dan R. Ghica, George Kaye, and David Sprunger. A compositional theory of digital circuits, 2022.

[10] Evan Patterson and other contributors. Algebraicjulia/catlab.jl: v0.12.2, May 2021. URL https://doi.org/10.5281/zenodo.4736069.

[11] Evan Patterson, Owen Lynch, and James Fairbanks. Categorical data structures for technical computing. *Compositionality*, 4:5, dec 2022. DOI: 10.32408/compositionality-4-5. URL https://doi.org/10.32408%2Fcompositionality-4-5.

[12] Mitchell Riley. Categories of optics, 2018.

[13] P. Selinger. A survey of graphical languages for monoidal categories. In *New Structures for Physics*, pages 289–355. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-12821-9_4. URL https://doi.org/10.1007%2F978-3-642-12821-9_4.

[14] Toby St. Clere Smithe. Bayesian updates compose optically, 2020.

[15] Paul Wilson and Fabio Zanasi. The cost of compositionality: A high-performance implementation of string diagram composition. 2021. DOI: 10.48550/ARXIV.2105.09257. URL https://arxiv.org/abs/2105.09257.

[16] Paul Wilson and Fabio Zanasi. Data-parallel algorithms for string diagrams, 2023. URL https://arxiv.org/abs/2305.01041.