Structural Operational Semantics for String Diagrammatic Languages

(Invited Talk Extended Abstract)*

Fabio Zanasi
University College London, UK
f.zanasi@ucl.ac.uk

Structural operational semantics ([34]) is a cornerstone of the theory of programming languages: by deriving the dynamics of a system from inference rules that follow the structure of its syntax, it offers a perspective on operational behaviour that is inherently *compositional*—the behaviour of a composite system ought to be a function of the behaviour of its components.

Turi and Plotkin's *bialgebraic semantics* (abstract GSOS) ([37, 27]) grounds structural operational semantics in category theory. Syntax is modelled as the algebra T_Σ of terms over a signature Σ, dynamics as a functor $\mathscr{F}$ fixing the transition type, (whose coalgebras $X \to \mathscr{F}(X)$ are the transition systems of interest), and a semantic specification as a certain kind of *distributive law* between syntax and dynamics. Any such specification induces a coalgebra $\beta : T_\Sigma \to \mathscr{F}(T_\Sigma)$, capturing the operational semantics on terms, while the final coalgebra Ω provides the denotational universe; the unique coalgebra morphism $[\![\cdot]\!]$ assigns to each term its behaviour:

$$\begin{array}{ccc} T_\Sigma & \overset{[\![\cdot]\!]}{\dashrightarrow} & \Omega \\ {\scriptstyle\beta}\downarrow & & \downarrow{\scriptstyle\omega} \\ \mathscr{F}(T_\Sigma) & \xrightarrow[\mathscr{F}[\![\cdot]\!]]{} & \mathscr{F}(\Omega) \end{array}$$

Crucially, the square lives in a category of algebras: $[\![\cdot]\!]$ is automatically a homomorphism, bisimilarity is a congruence, and compositionality of the semantics comes "for free". Syntactic rule formats guaranteeing these properties, such as De Simone's ([18]) and GSOS ([2]), arise as concrete presentations of distributive laws.

In the last few decades, *string diagrams* ([35, 33]) have emerged as a family of formal languages supporting algebraic reasoning about resource-sensitive systems and graphical formalisms, such as the circuits and networks found in quantum theory ([16, 15]), control theory ([13]), digital ([22]) and electrical ([7, 3]) circuit theory, concurrency theory ([5]), probabilistic computation ([20, 14, 25, **?**, 28]) and machine learning ([17]). What makes string diagrams appealing is their dual nature: like graphical models, they may be studied as combinatorial objects, with nodes and wires; like programming languages, they are a fully formal, two-dimensional syntax, whose terms are built from elementary components via sequential (;) and parallel ($\oplus$) composition, and identified up to the laws of symmetric monoidal categories. Rule-based definitions of the operational behaviour of string diagrammatic languages appear throughout this line of research—for instance, in the algebra of Span(Graph) ([26]), tile logic ([21]), the wire calculus ([36]), and the diagrammatic calculi of signal flow graphs and Petri nets ([5]). However,

*This extended abstract for EXPRESS/SOS 2026 is based on the journal paper [8], which extends the CONCUR paper [6]. Participation to EXPRESS/SOS 2026 is supported by ARIA's Safeguarded AI Programme.

Submitted to:
EXPRESS/SOS 2026

each of these examples was introduced on a case-by-case basis, in the absence of a general theory of operational semantics for diagrammatic languages.

Our works [8, 6] establish a bialgebraic perspective on string diagrams, endowing them with a structural operational semantics in the sense of Turi and Plotkin, and thereby providing a uniform foundation for these operational definitions. Adapting such a framework to string diagrams requires some technical work. The term-forming operations of string diagrams (sequential and parallel composition) are typed by the number of dangling wires on the left and right of the diagrams involved: syntax thus lives over *sorted* signatures—spans $\mathbb{N} \leftarrow \Sigma \rightarrow \mathbb{N}$ recording the two numbers for each generator in Σ—rather than sets. Moreover, since string diagrams are terms modulo the laws of symmetric monoidal categories, the algebra of diagrams is not a free monad but a quotient thereof, whose algebras are precisely the symmetric strict monoidal categories with the natural numbers as objects (also called *props*, see [29]). The distributive law giving a semantic specification must be quotiented as well. The end result is a discipline for specifying transition rules for the generators of a monoidal signature which guarantees, for every specification:

Well-definedness the transition relation is invariant under these laws, hence lives on string diagrams rather than terms;

Congruence bisimilarity is preserved by string diagrammatic operations (in particular, ; and $\oplus$);

Compositionality the denotational map $[\![\cdot]\!]$ is a homomorphism with respect to the two compositions: $[\![c\,;d]\!] = [\![c]\!]\,;[\![d]\!]$ and $[\![c \oplus d]\!] = [\![c]\!] \oplus [\![d]\!]$.

The same approach adapts to string diagrams for richer monoidal structures, such as hypergraph categories; interestingly, it does not extend to cartesian categories, where copying is incompatible with nondeterministic behaviour, as we discuss below.

As a proof of concept, the framework is instantiated on a versatile diagrammatic language Circ_R, called the resource calculus ([9, 11, 5]). Circ_R is parametrised by a semiring R: when R is a field, its diagrams model signal flow graphs, the circuits of linear time-invariant dynamical systems ([10, 1]); when $\mathsf{R} = \mathbb{N}$, the same syntax and rules yield a compositional account of Petri nets ([5]). Figure 1 shows the resulting operational semantics at work. A transition $c \xrightarrow[b]{a} d$ means that c may evolve to d while the values a are observed on the dangling wires on the left boundary of c, and b on those on the right: for instance, the register $-\overset{k}{\boxed{x}}-$ acts as a one-place buffer, emitting its content and storing the value it receives. The displayed run concerns the signal flow graph $fib_{k,l}$, which transforms the input stream $1\,0\,0\,0\ldots$ into the stream $1\,2\,3\,5\,8\ldots$ of Fibonacci numbers.

Moreover, the framework reveals which *further* equations on diagrams are compatible with an operational specification. Diagrammatic calculi typically identify diagrams up to additional equations, notably those of special Frobenius bimonoids, whose diagrams form hypergraph categories ([19, 4]). The calculus Circ_R has two such structures, a 'black' and a 'white' one: the black rules preserve the Frobenius equations over every semiring, whereas the white ones—read as addition—preserve them precisely when the labels form an Abelian group ([32]), matching the known equational theories of Circ_R over $\mathbb{R}$ ([11]) and over $\mathbb{N}$ ([5]). By contrast, the copying and discarding equations characterising cartesian syntax (Lawvere theories) ([24, 12]) are *incompatible* with the framework: nondeterministic behaviour cannot be copied—a single copied process resolves its choices once, while two independent copies choose independently—so the two sides of the copying law are not even trace equivalent.

A concluding observation ties the framework back to classical process calculi: the two operational readings of the Frobenius structure provide exactly the two classical synchronisation mechanisms. Consider the translation of a minimal calculus of communicating processes into string diagrams, in which

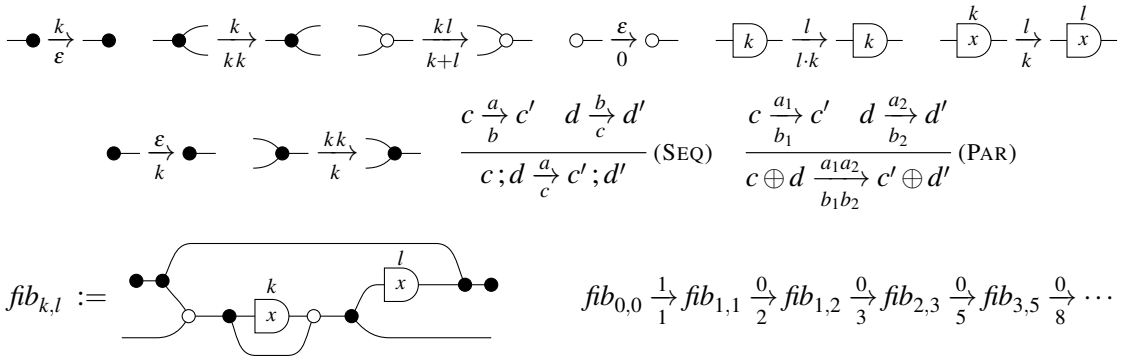

Figure 1: First two rows: operational semantics of $\mathsf{Circ}_{\mathsf{R}}$, with transition axioms for the generators and structural rules for the two compositions; k, l range over R, the words a, b, c over $\mathsf{R}^\star$, and ε is the empty word. The generators of $\mathsf{Circ}_{\mathsf{R}}$ not displayed are mirror images of those in the first row, with symmetric axioms; identities and symmetries admit the expected transitions. Last row: the signal flow graph $fib_{k,l}$, whose registers store the values k and l, and a run computing the Fibonacci sequence.

wires play the role of names: parallel composition shares names through the Frobenius comultiplication, and restriction caps the hidden wire. Under the black reading, all wires around a node must carry the same value: this yields synchronisation à la Hoare, as in CSP ([23]). Under the white reading, the values around a node add up, so that a send and a receive cancel out: this yields the handshake à la Milner of CCS and SCCS ([30, 31]). Therefore, the diagrammatic perspective reveals that the two classical synchronisation patterns are exactly the two operational readings of the same wiring structure, distinguished only by whether labels form an Abelian group.

References

[1] John Baez & Jason Erbele (2015): *Categories In Control*. *Theory and Applications of Categories* 30, pp. 836–881, doi:10.70930/tac/d5nq5dv8.

[2] Bard Bloom, Sorin Istrail & Albert R. Meyer (1995): *Bisimulation can't be traced*. *Journal of the ACM* 42(1), pp. 232–268, doi:10.1145/200836.200876.

[3] Guillaume Boisseau & Paweł Sobociński (2021): *String Diagrammatic Electrical Circuit Theory*. In: *Proceedings of the 4th International Conference on Applied Category Theory*, *EPTCS* 372, pp. 178–191, doi:10.4204/EPTCS.372.13.

[4] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński & Fabio Zanasi (2016): *Rewriting modulo symmetric monoidal structure*. In: *31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016)*, ACM, pp. 710–719, doi:10.1145/2933575.2935316.

[5] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2019): *Diagrammatic Algebra: from Linear to Concurrent Systems*. *Proceedings of the ACM on Programming Languages* 3(POPL), pp. 25:1–25:28, doi:10.1145/3290338.

[6] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2019): *Bialgebraic Semantics for String Diagrams*. In Wan J. Fokkink & Rob van Glabbeek, editors: *30th International Conference on Concurrency Theory (CONCUR 2019)*, *LIPIcs* 140, Schloss Dagstuhl, pp. 37:1–37:17, doi:10.4230/LIPIcs.CONCUR.2019.37.

[7] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2019): *Graphical Affine Algebra*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019)*, IEEE, pp. 1–12, doi:10.1109/LICS.2019.8785877.

[8] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2021): *Bialgebraic foundations for the operational semantics of string diagrams*. *Information and Computation* 281, p. 104767, doi:10.1016/j.ic.2021.104767.

[9] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2014): *A Categorical Semantics of Signal Flow Graphs*. In: *25th International Conference on Concurrency Theory (CONCUR 2014)*, *LNCS* 8704, Springer, pp. 435–450, doi:10.1007/978-3-662-44584-6_30.

[10] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2015): *Full Abstraction for Signal Flow Graphs*. In: *42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2015)*, ACM, pp. 515–526, doi:10.1145/2676726.2676993.

[11] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2017): *The Calculus of Signal Flow Diagrams I: Linear relations on streams*. *Information and Computation* 252, pp. 2–29, doi:10.1016/j.ic.2016.03.002.

[12] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2018): *Deconstructing Lawvere with distributive laws*. *Journal of Logical and Algebraic Methods in Programming* 95, pp. 128–146, doi:10.1016/j.jlamp.2017.12.002.

[13] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2021): *A Survey of Compositional Signal Flow Theory*. In: *Advancing Research in Information and Communication Technology: IFIP's Exciting First 60+ Years*, *IFIP Advances in Information and Communication Technology* 600, Springer, pp. 29–56, doi:10.1007/978-3-030-81701-5_2.

[14] Kenta Cho & Bart Jacobs (2019): *Disintegration and Bayesian inversion via string diagrams*. *Mathematical Structures in Computer Science* 29(7), pp. 938–971, doi:10.1017/S0960129518000488.

[15] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In: *35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, *LNCS* 5126, Springer, pp. 298–310, doi:10.1007/978-3-540-70583-3_25.

[16] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.

[17] Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul W. Wilson & Fabio Zanasi (2024): *Deep Learning with Parametric Lenses*. *CoRR* abs/2404.00408, doi:10.48550/arXiv.2404.00408.

[18] Robert De Simone (1985): *Higher-level synchronising devices in Meije-SCCS*. *Theoretical Computer Science* 37, pp. 245–267, doi:10.1016/0304-3975(85)90093-3.

[19] Brendan Fong & David I. Spivak (2019): *Hypergraph Categories*. *Journal of Pure and Applied Algebra* 223(11), pp. 4746–4777, doi:10.1016/j.jpaa.2019.02.014.

[20] Tobias Fritz (2020): *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics*. *Advances in Mathematics* 370, p. 107239, doi:10.1016/j.aim.2020.107239.

[21] Fabio Gadducci & Ugo Montanari (2000): *The tile model*. In: *Proof, Language and Interaction: Essays in Honour of Robin Milner*, MIT Press, pp. 133–166, doi:10.7551/mitpress/5641.003.0010.

[22] Dan R. Ghica, Achim Jung & Aliaume Lopez (2017): *Diagrammatic Semantics for Digital Circuits*. In: *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, *LIPIcs* 82, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 24:1–24:16, doi:10.4230/LIPIcs.CSL.2017.24.

[23] C. A. R. Hoare (1985): *Communicating Sequential Processes*. Prentice-Hall.

[24] Martin Hyland & John Power (2007): *The Category Theoretic Understanding of Universal Algebra: Lawvere Theories and Monads*. In: *Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin*, *ENTCS* 172, Elsevier, pp. 437–458, doi:10.1016/j.entcs.2007.02.019.

[25] Bart Jacobs, Aleks Kissinger & Fabio Zanasi (2021): *Causal inference via string diagram surgery: A diagrammatic approach to interventions and counterfactuals*. *Mathematical Structures in Computer Science* 31(5), pp. 553–574, doi:10.1017/S096012952100027X.

[26] Piergiulio Katis, Nicoletta Sabadini & Robert F. C. Walters (1997): *Span(Graph): an algebra of transition systems*. In: *Algebraic Methodology and Software Technology (AMAST 1997)*, *LNCS* 1349, Springer, pp. 322–336, doi:10.1007/BFb0000479.

[27] Bartek Klin (2011): *Bialgebras for structural operational semantics: an introduction*. *Theoretical Computer Science* 412(38), pp. 5043–5069, doi:10.1016/j.tcs.2011.03.023.

[28] Antonio Lorenzin & Fabio Zanasi (2025): *Bayesian Networks, Markov Networks, Moralisation, Triangulation: a Categorical Perspective*. *CoRR* abs/2512.09908, doi:10.48550/arXiv.2512.09908.

[29] Saunders Mac Lane (1965): *Categorical Algebra*. *Bulletin of the American Mathematical Society* 71, pp. 40–106, doi:10.1090/S0002-9904-1965-11234-4.

[30] Robin Milner (1982): *A Calculus of Communicating Systems*. Springer.

[31] Robin Milner (1983): *Calculi for synchrony and asynchrony*. *Theoretical Computer Science* 25(3), pp. 267–310, doi:10.1016/0304-3975(83)90114-7.

[32] Dusko Pavlovic (2009): *Quantum and classical structures in nondeterministic computation*. In: *Quantum Interaction (QI 2009)*, *LNCS* 5494, Springer, pp. 143–157, doi:10.1007/978-3-642-00834-4_13.

[33] Robin Piedeleu, Mateo Torres-Ruiz, Alexandra Silva & Fabio Zanasi (2025): *A Complete Axiomatisation of Equivalence for Discrete Probabilistic Programming*. In: *ESOP (2)*, *Lecture Notes in Computer Science* 15695, Springer, pp. 202–229, doi:10.1007/978-3-031-91121-7_9.

[34] Robin Piedeleu & Fabio Zanasi (2025): *An Introduction to String Diagrams for Computer Scientists*. Elements in Applied Category Theory, Cambridge University Press, doi:10.1017/9781009625715.

[35] Gordon D. Plotkin (2004): *A structural approach to operational semantics*. *Journal of Logic and Algebraic Programming* 60-61, pp. 17–139, doi:10.1016/j.jlap.2004.05.001.

[36] Peter Selinger (2011): *A survey of graphical languages for monoidal categories*. In: *New Structures for Physics*, *Lecture Notes in Physics* 813, Springer, pp. 289–355, doi:10.1007/978-3-642-12821-9_4.

[37] Paweł Sobociński (2009): *A non-interleaving process calculus for multi-party synchronisation*. In: *2nd Interaction and Concurrency Experience (ICE 2009)*, *EPTCS* 12, pp. 87–98, doi:10.4204/EPTCS.12.6.

[38] Daniele Turi & Gordon Plotkin (1997): *Towards a mathematical operational semantics*. In: *12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997)*, IEEE, pp. 280–291, doi:10.1109/LICS.1997.614955.