Full Abstraction for Signal Flow Graphs

Filippo Bonchi
ENS Lyon, U. Lyon, CNRS, INRIA

Paweł Sobociński
U. Southampton, UK

Fabio Zanasi
ENS Lyon, U. Lyon, CNRS, INRIA

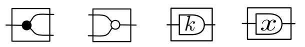
We would be glad to participate to ICE 2015 by announcing our POPL 2015 paper *Full Abstraction for Signal Flow Graphs*. We believe that this work is relevant for ICE in many respects, among which we highlight two: (1) the introduced calculus is closely related to several *algebras of connectors* studied by Bruni, Lanese, Melgratti and Montanari and, somehow, it can be regarded as a descendant of the *wire calculus* introduced by the second author during the second edition of ICE; (2) it provides an highly innovative approach to the semantics of programming languages showing some concrete benefits in dropping any notion of *causality*.

Below, we report the Introduction of the original paper and we refer the interested reader to the proceedings of POPL 2015 for the full version.

[T]he reason why physics has ceased to look for causes is that in fact there are no such things. The law of causality, I believe, like much that passes muster among philosophers, is a relic of a bygone age, surviving, like the monarchy, only because it is erroneously supposed to do no harm.

B. Russel 1913

Introduction. Signal flow graphs (SFGs) are foundational structures in control theory and signal processing studied since at least the 1950s [21]. They can be constructed from small set of basic components (displayed below) and feedbacks.



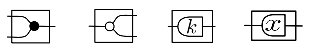
(1)

Signals, which take values over a field k, flow from left to right. The leftmost component *duplicates* the signal, the second *sums* the two signals arriving on the left and the third *multiplies* the signal by a scalar $k \in \mathsf{k}$. The rightmost one is a *delay*: when a sequence of signals $k_0, k_1, k_2, \dots$ arrives on the left, it outputs the sequence $0, k_0, k_1 \dots$ It can thus be thought as a synchronous one place buffer initialised with 0.

A simple mathematical meaning can be given to those SFGs where feedbacks pass through (at least) one delay component. It is well known (see e.g. [19]) that SFGs with this restriction, one input and one ouput port denote so-called rational linear functions. In traditional approaches, however, SFGs are not treated as interesting mathematical structures per se: formal analyses typically mean the introduction of latent variables and translations into systems of linear equations— although, more recently, they have also attracted the use of coalgebraic tools [24, 3]. Our work, instead, follows the series of recent papers [6, 5, 2, 12, 28] where SFGs are understood as structures known as *string diagrams* and studied as mathematical objects of interest in their own right—this approach is known as *network theory* [1]. The majority of the attention so far has been focused on what we call the *denotational semantics*: differently from the classical approach, string diagrams, in general, give rise to *linear relations* rather than functions.

Submitted to:
ICE 2015- Short Announcements

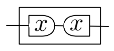
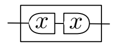
The string diagrams that are considered are not restricted by any side conditions on feedbacks, being all those diagrams generated from the basic components (1), together with their duals:


(2)

Intuitively, in (2) the signal flows from right to left. This means that diagrams constructed using both components in (1) and (2) have no univocal flow direction and require a relational model.

Network theory brings fundamentally new ingredients to the field of signal flow graphs. First, the relational semantics is a *compositional* account of their behavior that enjoys a sound and complete axiomatisation independently discovered in [6, 5] and [2]. Second, the axiomatisation has uncovered a rich underlying mathematical playground – featuring two Hopf algebras and two Frobenius algebras – which also reveals connections with quantum phenomena [4, 28, 2]. Third, it has resulted in a subtle re-evaluation of *causality* as a central ingredient of SFGs. In 1953 Mason [21] wrote: “flow graphs differ from electrical network graphs in that their branches are directed. In accounting for branch directions it is necessary to take an entirely different line of approach from that adopted in electrical network topology.” Instead, our results suggest that *direction of signal flow is not a primitive notion*: this argument has been made informally already in [5, 2] but is rigorously shown in our work. Similar ideas are prominent in the *behavioural approach* in control theory [27].

In this work, we introduce *operational semantics* to the network theoretic accounts of signal flow graphs: we show that string diagrams can be thought of as terms of a process calculus and executed as state machines. For this reason we shall call our string diagrams *circuit diagrams* or simply *circuits*. Reconciling the operational perspective with the established denotational model turns out to be quite subtle. Indeed, the denotational semantics is in a sense too abstract: finite computations that reach *deadlocks* are ignored. Such deadlocks can arise for instance when components of (1) are composed with the those of (2) and, intuitively, the signal flows from the left and right toward the middle. For an example, consider the circuit below on the left.



(3)

In a first step, the signals arriving from left and right are stored in the two buffers. Then, the stored values are compared in the middle of the circuit: if they do not agree then the computation gets stuck. The circuit on the right features another problem, which we call *initialisation*. Intuitively, the flow goes from the middle toward left and right. All its computations are forced to start by emitting on the left and on the right the value 0 which is initially stored in the two buffers. The two circuits are denotationally equivalent, but their operational behaviour can be obviously distinguished: the leftmost does not have initialisation and the rightmost cannot deadlock.

Deadlock and initialisation are dual problems at the heart of the mismatch of operational and denotational semantics. We show that circuits in *cospan form*, namely circuits built from components in (1) followed by those in (2) (like the leftmost circuit in (3)), are free from initialisation. Instead, circuits in *span form*, i.e., those built from components in (2) followed by (1) (like the rightmost in (3)) are free from deadlock. This is interesting because the equational theory developed in [5, 2] asserts that any circuit is equivalent to both one in cospan and one in span form. This duality of deadlock and initialisation helps us in proving a *full abstraction* result: for those circuits that are free from both deadlock and initialisation, the operational and the denotational semantics agree.

Our second main theorem is a *realisability* result: for any denoted behaviour there exists some circuit that properly, without deadlocks or initialisation, *realises* it. The key for the proof is the fact that any circuit in [5, 2] is equivalent (according to the denotational semantics) to a signal flow graph up to some "rewiring". This result allow us to impose a syntactic restriction to our circuits to guarantee deadlock and initialisation freedom. By virtue of the full abstraction results we can thus safely use the axiomatization of [5, 2] to reason about the operational behaviour of these circuits.

Summarising, the main results of our work are:

- a structural operational semantics for the network-theoretic account of SFGs;
- a full abstraction theorem relating the operational and the denotational semantics previously introduced in [5, 2];
- a realisability theorem: every behaviour can be implemented by a circuit without deadlock and initialisation;
- a formal explanation of the fact that direction of flow is a derivative notion.

Related work. String diagrams originally came to the fore in the study of monoidal categories because they clear away swathes of cumbersome coherence bureaucracy, thereby dramatically simplifying algebraic arguments: in particular, they are useful for characterising *free* monoidal categories [16, 14, 25].

In this work we consider particular symmetric monoidal categories, called PROPs [20, 17] (PROduct-and-Permutation categories). PROPs are a useful setting for the study of string diagrams and especially monoidal theories—Lack's theory of composing PROPs [17] was used to derive the axiomatisation in [6, 5]. PROPs have also recently been used by computer scientists: Lafont's study of boolean circuits [18], Bruni, Montanari, Plotkin, and Terreni [8] have used them to give an alternative presentation of Milner's bigraphs while Fiore and Campos [11] presented a theory of directed acyclic graphs. Our operational semantics is related to Katis, Sabadini and Walters' [15] Span(Graph) algebra of transition systems and the algebra of connectors of Bruni, Lanese and Montanari [7]. String diagrams are increasingly used by computer scientists: for instance we mention Pavlovic's monoidal computer [22, 23] where they are employed to study classical notions of computability and computational complexity.

The interplay of Hopf algebras and Frobenius algebras, at the core of the axiomatisation of the denotational semantics, appeared first in the work of Coecke, Duncan and Kissinger [9, 10] on the ZX-calculus, used in the study of quantum circuits. Similar algebraic interactions emerged in the study of Petri nets [26] and in string-diagrammatic theories of asynchronous circuits [13].

References

[1] John C. Baez (2014): *Network Theory*. `http://math.ucr.edu/home/baez/networks/`.

[2] John C. Baez & Jason Erbele (2014): *Categories In Control*. *CoRR* abs/1405.6881. Available at `http://arxiv.org/abs/1405.6881`. `http://arxiv.org/abs/1405.6881`.

[3] Henning Basold, Marcello Bonsangue, Helle H. Hansen & Jan Rutten (2014): *(Co)Algebraic Characterizations of Signal Flow Graphs*. In: *To appear in LNCS*.

[4] F. Bonchi, P. Sobociński & F. Zanasi (2014): *Interacting Bialgebras are Frobenius*. In: *FoSSaCS '14*, Springer.

[5] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2014): *A Categorical Semantics of Signal Flow Graphs*. In: *CONCUR*.

[6] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2014): *Interacting Hopf Algebras*. *CoRR* abs/1403.7048. Available at `http://arxiv.org/abs/1403.7048`. `http://arxiv.org/abs/1403.7048`.

[7] Roberto Bruni, Ivan Lanese & Ugo Montanari (2006): *A basic algebra of stateless connectors*. *Theor Comput Sci* 366, pp. 98–120.

[8] Roberto Bruni, Ugo Montanari, Gordon Plotkin & Daniele Terreni: *On hierarchical graphs: reconciling bigraphs, gs-monoidal theories and gs-graphs .*

[9] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In: *ICALP'08*, pp. 298–310.

[10] Bob Coecke, Ross Duncan, Aleks Kissinger & Quanlong Wang (2012): *Strong Complementarity and Non-locality in Categorical Quantum Mechanics*. In: *LiCS'12*, pp. 245–254.

[11] Marcelo P. Fiore & M. Devesas Campos (2013): *The Algebra of Directed Acyclic Graphs*. In: *Abramsky Festschrift*, *LNCS* 7860.

[12] Brendan Fong (2013): *A compositional approach to control theory*. PhD Transfer Report.

[13] Dan R. Ghica (2013): *Diagrammatic Reasoning for Delay-Insensitive Asynchronous Circuits*. In: *Abramsky Festschrift*, pp. 52–68. Available at `http://dx.doi.org/10.1007/978-3-642-38164-5_5`.

[14] Andre Joyal & Ross Street (1991): *The Geometry of Tensor Calculus,* I. *Adv. Math.* 88, pp. 55–112.

[15] Piergiulio Katis, Nicoletta Sabadini & Robert Frank Carslaw Walters (1997): *Span(Graph): an algebra of transition systems*. In: *AMAST '97*, Springer, pp. 322–336.

[16] G. M. Kelly & M. L. Laplaza (1980): *Coherence for compact closed categories*. *J. Pure Appl. Algebra* 19, pp. 193–213.

[17] Stephen Lack (2004): *Composing PROPs*. *Theor App Categories* 13(9), pp. 147–163.

[18] Yves Lafont (2003): *Towards an algebraic theory of Boolean circuits*. *J Pure Appl Alg* 184, pp. 257–310.

[19] B.P. Lahti (1998): *Signal Processing and Linear Systems*. Oxford University Press.

[20] Saunders Mac Lane (1965): *Categorical Algebra*. *B Am Math Soc* 71, pp. 40–106.

[21] Samuel J Mason (1953): *Feedback Theory: I. Some Properties of Signal Flow Graphs*. MIT Research Laboratory of Electronics.

[22] Dusko Pavlovic (2013): *Monoidal computer I: Basic computability by string diagrams*. *Inf. Comput.* 226, pp. 94–116. Available at `http://dx.doi.org/10.1016/j.ic.2013.03.007`.

[23] Dusko Pavlovic (2014): *Monoidal computer II: Normal complexity by string diagrams*. *CoRR* abs/1402.5687. Available at `http://arxiv.org/abs/1402.5687`.

[24] Jan J. M. M. Rutten (2005): *A tutorial on coinductive stream calculus and signal flow graphs*. *Theor. Comput. Sci.* 343(3), pp. 443–481. Available at `http://dx.doi.org/10.1016/j.tcs.2005.06.019`.

[25] Peter Selinger (2009): *A survey of graphical languages for monoidal categories*. ArXiv:0908.3347v1 [math.CT].

[26] Paweł Sobociński (2013): *Nets, relations and linking diagrams*. In: *CALCO '13*.

[27] Jan C Willems (2007): *The behavioural approach to open and interconnected systems*. *IEEE Contr. Syst. Mag.* 27, pp. 46–99.

[28] W. J. Zeng & Jamie Vicary (2014): *Abstract structure of unitary oracles for quantum algorithms*. *CoRR* abs/1406.1278.