Graphical Resource Algebra, from Linear to Affine

Filippo Bonchi[1], Robin Piedeleu[2], Paweł Sobociński[3], and Fabio Zanasi[2]

[1]Universitá di Pisa, Italy

[2]University College London, UK

[3]University of Southampton, UK

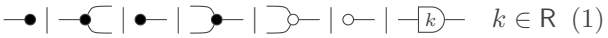
Graphical linear algebra is a diagrammatic language that allows compositional reasoning about R-linear systems, for different semirings R. When R is a field, its semantics and equational theory are well-understood. In this context, graphical linear algebra is closely related to signal flow graphs, another calculus commonly used in engineering and control theory to specify linear dynamical systems.

Recent work by the authors [4] revealed that when $\mathsf{R} = \mathbb{N}$, graphical linear algebra can be interpreted as an algebra of connectors that manipulate finite discrete resources. When endowed with state, the resulting calculus is expressive enough to capture the behaviour of Petri nets in a natural way and allows for modular reasoning about their semantics.

In an orthogonal direction, we showed how to extend this formalism with a connector for *affine* behaviour [5]. The extension, which we call graphical affine algebra, is simple but remarkably powerful: it can model systems with richer patterns of behaviour such as mutual exclusion–when $\mathsf{R} = \mathbb{N}$—or non-passive electrical components—when $\mathsf{R} = \mathbb{R}(x)$.

Our main technical contributions are complete axiomatisations of graphical linear/affine algebra for all of the above interpretations. We have also shown, as case studies, how the same language can capture electrical circuits and the calculus of stateless connectors—a coordination language for distributed systems.

Graphical linear algebra (GLA) [8, 16] is a diagrammatic language used to reason compositionally about different types of linear computing devices. String diagrams of GLA are sequential and parallel compositions of the following basic operations, for some semiring R.



Here ⊃○— represents addition, ○— the constant zero, —[k]— is multiplication by k, —●⊂ copy, —● discard, while ⊃●— and ●— are the same operations right-to-left. This semantics is formalised via a recursively defined functor from diagrams to *relations* over R-vectors: thus right-to-left operations are simply denoted by the opposite relations of their left-to-right cousins.

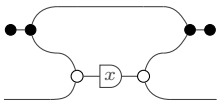
Over fields, GLA has some claims of being fundamental. For $\mathsf{R} = \mathbb{Z}/2$, it is the syntax of the phase-free ZX-calculus [7], a simple algebra for pure state qubit computation [10]. For $\mathsf{R} = \mathbb{R}(x)$ the field of polynomial fractions, it provides a compositional account of signal flow graphs [14, 13, 15], a graphical language commonly used by engineers to specify linear dynamical systems. In fact, GLA generalises signal flow graphs [2, 6, 12] and provides a purely graphical framework in which to recast well-known control-theoretic notions.

1 From Control to Concurrency Theory

More recently, in [4], we showed that, GLA over $\mathbb{N}$ captures concurrent patterns of interaction, including, for instance, a compositional

account of the semantics of Petri nets. Indeed, the lack of additive inverses in $\mathbb{N}$ is well-suited to situations where negative resources (here, the tokens in Petri nets) are not meaningful.

In order to explain this difference, consider the following instructive example:



(2)

In this context, the scalar —[x]— plays the distinguished role of a *register*: it admits an operational interpretation as a simple buffer that holds the value it last observed on the left for one timestep before releasing it on the right. This is crucial to capture stateful systems. Over $\mathbb{R}(x)$, the behaviour of (2), which can be computed from the denotation of the generators, is trivial: it is equal to the full relation —• •— $= \mathbb{R}(x) \times \mathbb{R}(x)$, relating any input to any output.

The "Aha-Erlebnis" is that, once interpreted over $\mathbb{N}$, the same diagram with the same operational understanding of the register models a non-trivial behaviour. Wires now carry discrete tokens that cannot be borrowed (i.e. they cannot be a negative quantity). Thus, when computing the denotation of the diagram in (2), the output $n \in \mathbb{N}$ is now forced to be a number of tokens smaller than the one $k \in \mathbb{N}$ stored in the register, and the new value in the register will be $m + (k - n)$, for input $m \in \mathbb{N}$. In other words, the diagram now models the same behaviour as a *place* in a Petri net!

Note that, in these different computational interpretations—from quantum and control-theoretic to concurrent—the set (1) of syntactic primitives and the specification of their relational behaviour remains the same. What changes is the denotational domain, that is, the kinds of relations that are characterised. GLA over a field (e.g. $\mathbb{Z}/2$, $\mathbb{R}$ or $\mathbb{R}(x)$), equipped with the equations of *interacting Hopf algebras*, axiomatises *linear* relations (relations that are linear subspaces) [8, 16]. Sound and fully complete axiomatisations for GLA over a field were given independently in [8, 16] and [2]. One of our main technical contributions is a sound and fully complete axiomatisation of GLA over $\mathbb{N}$, complete for *additive* relations (relations that are $\mathbb{N}$-semimodules, i.e. ,containing the zero vector and closed under addition) [4].

To showcase expressiveness, we have also shown that the behaviour of Petri nets is definable in GLA in a natural way. In the spirit of process algebra, the approach is compositional, allowing for the representation of open systems and emphasising interaction; in the spirit of Petri, the syntax is graphical, emphasising the connection topology of systems.

2 Richer Behaviour: Affine Relations

Concurrent programming can be seen as the marriage of parallelism with synchronisation mechanisms. One of the earliest and most influential synchronisation mechanism is *mutual exclusion* [11], with the same underlying idea present in modern concurrent programming through hardware-assisted atomic constructs such as *compare-and-set* (CAS). Any theory that takes up Abramsky's challenge [1] to identify the fundamental structures of concurrency ought to be expressive enough to account for such fundamental synchronisation patterns.

GLA over $\mathbb{N}$ is not quite expressive enough to capture essential behaviour patterns such as mutual exclusion. Indeed, consider the following idealised mutual exclusion connector, as considered in the *calculus of stateless connectors* [9].

(3)

The legal behaviours, as an $\mathbb{N}$-valued relation, is the finite set $\{(\binom{0}{0}, 0), (\binom{1}{0}, 1), (\binom{0}{1}, 1)\}$ indicating that only one of the two inputs can synchronise with the output at any one time. The

relation is not additive: e.g. $(\binom{1}{0}, 1) + (\binom{0}{1}, 1) = (\binom{1}{1}, 2)$ is *not* included. We shall see, however, that it is an $\mathbb{N}$-*affine* relation, in a sense made precise in [5]. Moving from additive to affine relations expands the relational universe. For example, the empty relation $\varnothing$ is affine, but not additive.

The concept of affinity is of course better known over fields and is the mathematical playground of affine and convex geometry. It turns out that moving to affine relations is fruitful also in this context, in modelling electrical circuits for example. GLA has already been used to define a compositional semantics for *passive* linear circuits [3], that is, electrical circuits built exclusively from resistors, inductors and capacitors. The advantage of this approach is that it provides a rigorous setting in which to perform *open* network analysis purely diagrammatically.

Non-passive components, however, like voltage and current sources, are not linear but affine. For example, a k-volt source constrains the voltage (ϕ_1, ϕ_2) and current (i) pairs to be the following relation:

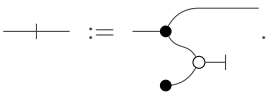
$$\longmapsto \quad \left\{ \binom{\phi_1}{i}, \binom{\phi_2}{i} \right) \mid \phi_2 - \phi_1 = k \} \qquad (4)$$

In [5], we have shown that *(i)* the syntax of GLA can be extended in a simple and principled fashion to capture $\mathbb{N}$-affine and $\mathbb{R}$-affine relations and *(ii)* have given an equational characterisation of denotational equality, arriving at a sound and complete calculus for both types of affine relations.

Extending GLA: Graphical Affine Algebra. The syntax of Graphical Affine Algebra (GAA) extends (1) with just one additional connector:

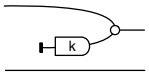
expressing the constant '1' behaviour and representing the relation $\{(\bullet, 1)\}$. This simple extension allows us to capture, not only the aforementioned examples, but all $\mathbb{N}$ or $\mathbb{R}$-affine relations—in fact, affine relations over any field $\mathbb{K}$.

For mutual exclusion, we first define $\dashv$ as



The result is a wire that can only carry 0 or 1 . Given this, the mutual exclusion connector (3) is simply the composition of $\supset$ with $\dashv$.

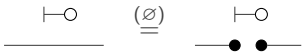
For non-passive circuit components, it is not difficult to see that the relation in (4) is expressed by the following GAA diagram:



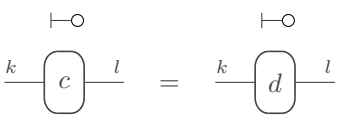
Finally, the empty relation $\varnothing$—which is both $\mathbb{N}$- and $\mathbb{R}$-affine—appears in our syntax as $\vdash\!\circ$ Here the new $\vdash$ generator is composed with $-\!\circ$. This amounts to asserting "$1 = 0$"; denotationally, this is the composition of $\{(\bullet, 1)\}$ with $\{(0, \bullet)\}$, which is empty.

Equational characterisations. As our last main technical contribution, we provide two sound and fully complete axiomatisations for GAA over affine relations, for the two cases of interest in our applications: $\mathbb{N}$ and any field $\mathbb{K}$.

The equational theories are simple, with only a few additional equations that govern the interaction of $\vdash$ with the remaining GLA primitives. A particularly interesting equation is shared by the two theories and concerns the properties of the empty relation—it allows us to disconnect any wire:



This guarantees that $\vdash\!\circ$ behaves analogously to logical false; in particular, we are able to prove that



for any diagrams c and d, in accordance with the semantics.

References

[1] Samson Abramsky. What are the fundamental structures of concurrency? we still don't know! *arXiv:1401.4973*, 2014.

[2] John Baez and Jason Erbele. Categories in control. *TAC 30:836–881*, 2015.

[3] John C. Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. *arXiv:1707.08321*, 2017.

[4] Filippo Bonchi, Josh Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. In *POPL '19*. Available at http://www.zanasi.com/fabio/files/paperPOPL19.pdf.

[5] Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Graphical affine algebra. In *LICS*, 2019. Available at http://www.zanasi.com/fabio/files/paperLICS19.pdf.

[6] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Full abstraction for signal flow graphs. In *POPL '15*, pages 515–526.

[7] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Interacting bialgebras are Frobenius. In *FOSSACS '14*, pages 351–365.

[8] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Interacting Hopf algebras. *J Pure Appl Alg*, 221(1):144–184, 2017.

[9] Roberto Bruni, Ivan Lanese, and Ugo Montanari. A basic algebra of stateless connectors. *Theor Comput Sci*, 366(1–2):98–120, 2006.

[10] Bob Coecke and Ross Duncan. Interacting quantum observables. In *ICALP '08*, pages 298–310.

[11] Edsger W. Dijkstra. Cooperating sequential processes. In *The origin of concurrent programming*, pages 65–138. Springer, 1968.

[12] Brendan Fong, Paolo Rapisarda, and Paweł Sobociński. A categorical approach to open and interconnected dynamical systems. In *LICS '16*.

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

[14] Claude E. Shannon. The theory and design of linear differential equation machines. Technical report, National Defence Research Council, 1942.

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

[16] Fabio Zanasi. *Interacting Hopf Algebras: the theory of linear systems*. PhD thesis, Ecole Normale Supérieure de Lyon, 2015.