Diagrammatic Algebra: from Linear to Concurrent Systems

Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński and Fabio Zanasi

PoPL '19, Cascais, Portugal

Question

What are the Fundamental Structures of Concurrency? We still don't know!

Samson Abramsky [1,2]

Oxford University Computing Laboratory
Oxford, U.K.

Abstract

Process algebra has been successful in many ways; but we don't yet see the lineaments of a fundamental theory. Some fleeting glimpses are sought from Petri Nets, physics and geometry.

Keywords: Concurrency, process algebra, Petri nets, geometry, quantum information and computation.

Electronic Notes in Theoretical Computer Science, 2006

Process algebras vs. Petri nets

In this talk

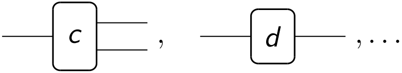
Towards bridging the gap between the two approaches:

- Start from a compositional diagrammatic language for linear dynamical systems.
- Give it a *resource-conscious semantics* by changing the domain from a field to the semiring $\mathbb{N}$.
- Provide a sound and complete equational theory for this new semantics.
- Showcase the expressiveness of the calculus by embedding Petri nets with their usual operational semantics.

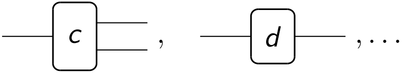
Section 1

A simple graphical language

Drawing open systems

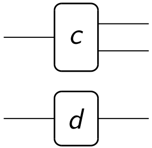


Drawing open systems

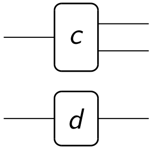


$$\llbracket c \rrbracket_X \subseteq X \times X^2,\ \llbracket d \rrbracket_X \subseteq X \times X \ldots \text{ for some fixed set } X.$$

Parallel composition



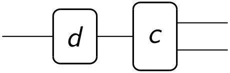
Parallel composition



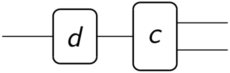
$$\mapsto$$

$$\left\{ \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \right) \;\middle|\; \left(x_1, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right) \in [\![\, c \,]\!]_X \,, (x_2, y_3) \in [\![\, d \,]\!]_X \right\}$$

Synchronising composition



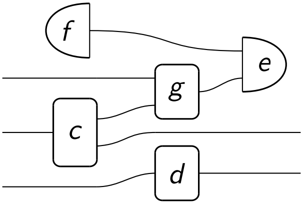
Synchronising composition



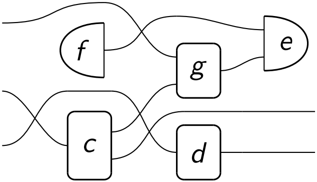
$$\mapsto$$

$$\left\{ \left(x, \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right) \;\middle|\; \exists y\, (x, y) \in [\![\, d \,]\!]_X \,, \left(y, \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right) \in [\![\, c \,]\!]_X \right\}$$

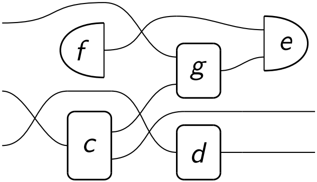
More complex networks



Only the connectivity matters

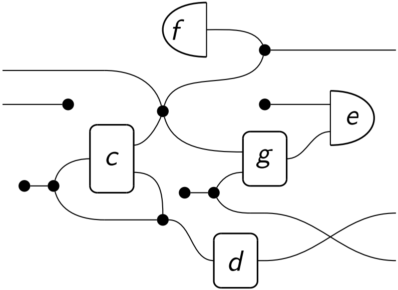


Only the connectivity matters

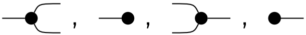


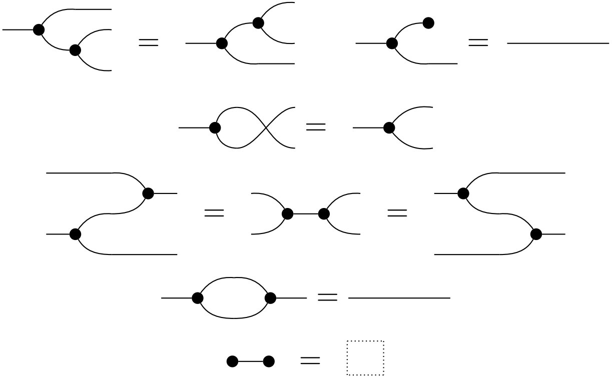
$$\left[\!\left[\quad \right]\!\right]_X = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix} \right) \;\middle|\; x, y \in X \right\}$$

Multiple connections



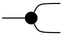
Frobenius monoids

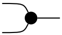
Special boxes/systems:  satisfying:



- form a *special commutative Frobenius monoid*.

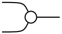
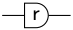
Interpreted as:

$[\![$  $]\!]_X = \left\{ \left(x, \begin{pmatrix} x \\ x \end{pmatrix} \right) \mid x \in X \right\}$ $\quad$ $[\![$  $]\!]_X = \{(x, \bullet) \mid x \in X\}$

$[\![$  $]\!]_X = \left\{ \left(\begin{pmatrix} x \\ x \end{pmatrix}, x \right) \mid x \in X \right\}$ $\quad$ $[\![$  $]\!]_X = \{(\bullet, x) \mid x \in X\}$

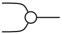
More algebraic structure

If $X = \mathsf{R}$ is a semiring we buy ourselves more structure:

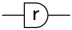
,  and 

for $r \in R$

$$\downarrow$$

$[\![$  $]\!]_{\mathsf{R}} = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, x + y \right) \mid (x, y) \in \mathsf{R}^2 \right\}$

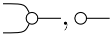
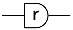
$[\![$  $]\!]_{\mathsf{R}} = \{(0, \bullet)\}$

$[\![$ 

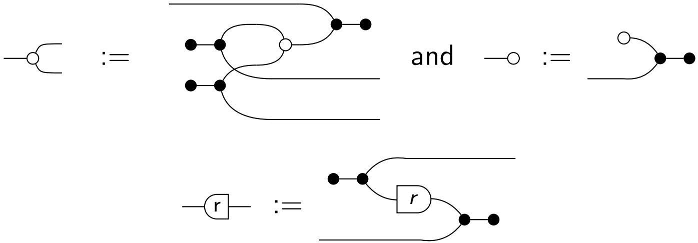
$]\!]_{\mathsf{R}} = \{(x, rx) \mid x \in \mathsf{R}\}$

More algebraic structure

If $X = \mathsf{R}$ is a semiring we buy ourselves more structure:

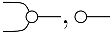
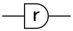
 and  for $r \in R$

and tranposes for free:

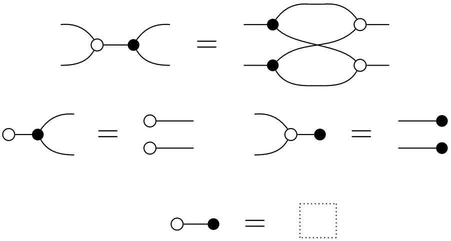


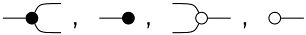
More algebraic structure

If $X = \mathrm{R}$ is a semiring we buy ourselves more structure:

 and  for $r \in R$

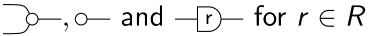
satisfying:



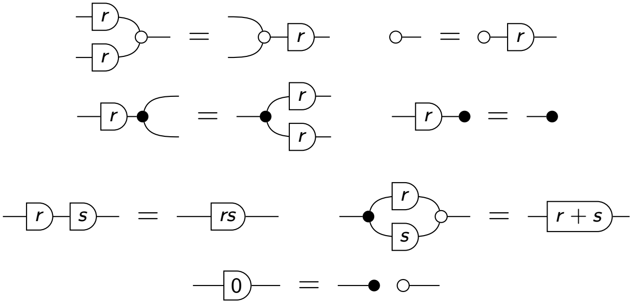
-  form a *bimonoid*.

More algebraic structure

If $X = \mathsf{R}$ is a semiring we buy ourselves more structure:



satisfying:



- Encode the additive and multiplicative operations of R.

Adding state

- Introduce $-\boxed{x}-$ that we interpret as a state-holding register.

Adding state

- Introduce $-\boxed{x}-$ that we interpret as a state-holding register.
- A *stateful* diagram $c\colon k \to l$ is interpreted as a relation

$$[\![\, c \,]\!] \subseteq \mathsf{R}^{s+k} \times \mathsf{R}^{s+l}$$

  where s is the number of $-\boxed{x}-$ in c.
- Semantics extended inductively with

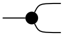
$$[\![-\boxed{x}-]\!]_{\mathsf{R}} = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix} \right) \,\middle|\, x, y \in \mathsf{R} \right\}$$

Section 2

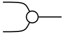
Detour: the Linear Interpretation

Linear relations

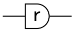
As relations over a field $\mathbb{K}$:

$[\![$  $]\!]_{\mathbb{K}} = \left\{ \left(x, \begin{pmatrix} x \\ x \end{pmatrix} \right) \mid x \in \mathbb{K} \right\}$

$[\![$  $]\!]_{\mathbb{K}} = \{ (x, \bullet) \mid x \in \mathbb{K} \}$

$[\![$  $]\!]_{\mathbb{K}} = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, x + y \right) \mid (x, y) \in \mathbb{K}^2 \right\}$

$[\![$  $]\!]_{\mathbb{K}} = \{ (0, \bullet) \}$

$[\![$  $]\!]_{\mathbb{K}} = \{ (x, rx) \mid x \in \mathbb{K} \}$

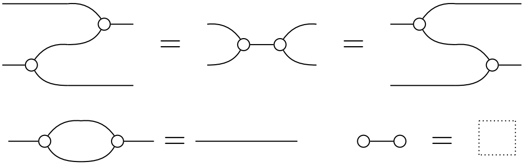
- For a diagram $c \colon k \to l$, $[\![c]\!]_{\mathbb{K}}$ is a linear subspace of $\mathbb{K}^k \times \mathbb{K}^l$, i.e., a relation closed under $\mathbb{K}$-linear combinations.

Complete equational theory

Interacting Hopf algebras

Filippo Bonchi [a], Paweł Sobociński [b], Fabio Zanasi [c,*]

- Addition is also a special commutative Frobenius monoid:



- Scalars are invertible:

for $r \neq 0$

Linear dynamical systems

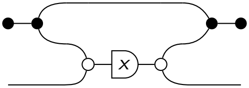
For the stateful linear case:

- $\mathbb{K} = \mathbb{R}(x)$ susbumes the notion of state we introduced.
- Semantics in terms of generalised *streams* (Laurent series).
- Models *linear dynamical systems* (e.g., filters, amplifiers)
- Generalisation of Shannon's *signal flow graphs*.
- Reformulates control-theory in diagrammatic terms (e.g., controllability, observability).

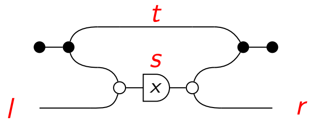
Section 3

The Resource Interpretation

Motivating example

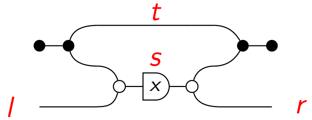


Motivating example



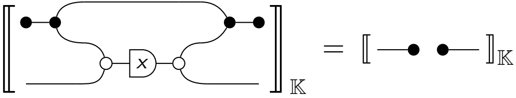
$$\mapsto \left\{ \left(\begin{pmatrix} s \\ l \end{pmatrix}, \begin{pmatrix} s' \\ r \end{pmatrix} \right) \,\middle|\, \exists t \quad \begin{matrix} s = t + r, \\ s' = t + l \end{matrix} \right\}$$

Motivating example

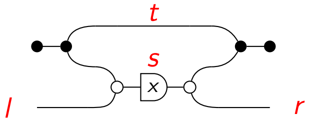


$$\mapsto \left\{ \left(\begin{pmatrix} s \\ l \end{pmatrix}, \begin{pmatrix} s' \\ r \end{pmatrix} \right) \;\middle|\; \exists t \quad \begin{matrix} s = t + r, \\ s' = t + l \end{matrix} \right\}$$

Over a field $\mathbb{K}$, we can relate any two l and r:

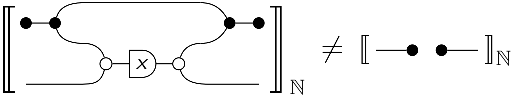


Motivating example

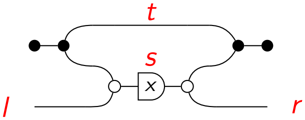


$$\mapsto \left\{ \left(\begin{pmatrix} s \\ l \end{pmatrix}, \begin{pmatrix} s' \\ r \end{pmatrix} \right) \;\middle|\; \exists t \;\; \begin{matrix} s = t + r, \\ s' = t + l \end{matrix} \right\}$$

Over $\mathbb{N}$, we must have $r \leq s$ so:

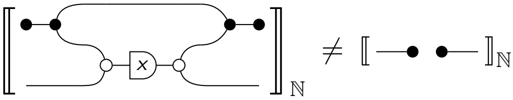


Motivating example



$$\mapsto \left\{ \left(\begin{pmatrix} s \\ l \end{pmatrix}, \begin{pmatrix} s' \\ r \end{pmatrix} \right) \;\middle|\; \exists t \;\; \begin{matrix} s = t + r, \\ s' = t + l \end{matrix} \right\}$$

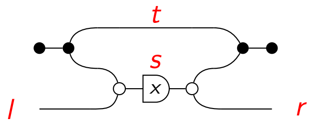
Over $\mathbb{N}$, we must have $r \leq s$ so:



Intuition

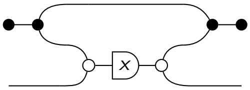
Without additive inverses, we cannot borrow arbitrary quantities.

Motivating example



$$\mapsto \left\{ \left(\begin{pmatrix} s \\ l \end{pmatrix}, \begin{pmatrix} s' \\ r \end{pmatrix} \right) \;\middle|\; \exists t \;\; \begin{matrix} s = t + r, \\ s' = t + l \end{matrix} \right\}$$

Over $\mathbb{N}$, we must have $r \leq s$ so:

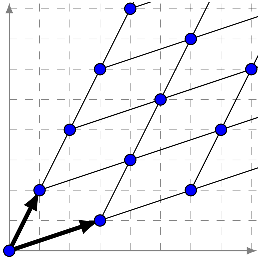
$[\![$ 

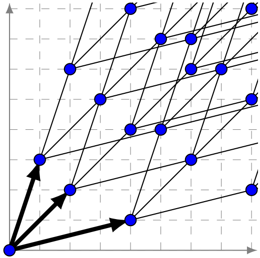
$]\!]_{\mathbb{N}} \neq [\![$  $]\!]_{\mathbb{N}}$

- This diagram behaves like the *place of a Petri net!*

Additive relations

For a diagram $c\colon k \to l$, $[\![c]\!]_{\mathbb{N}}$ is an *additive relation*: a finitely-generated submonoid of $\mathbb{N}^k \times \mathbb{N}^l$, i.e., a relation closed under addition and containing $(\mathbf{0}, \mathbf{0})$.





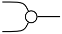
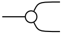
Proposition

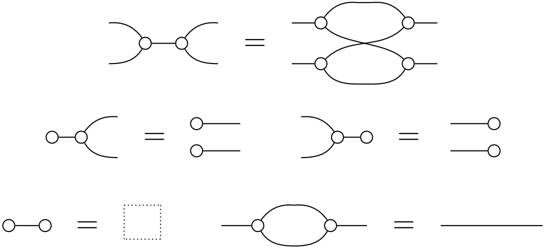
Finitely-generated additive relations form a symmetric monoidal category, AddRel.

- The proof that they compose is non-trivial and relies on Dickson's lemma.

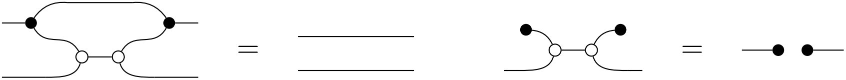
Complete equational theory

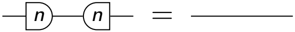
The Resource Calculus (RC) := every equation in Section I +

-  ,  and  ,  form an idempotent bimonoid:



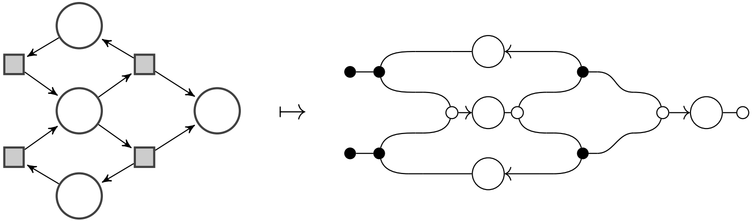
- And



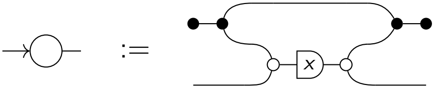


for $n \neq 0$

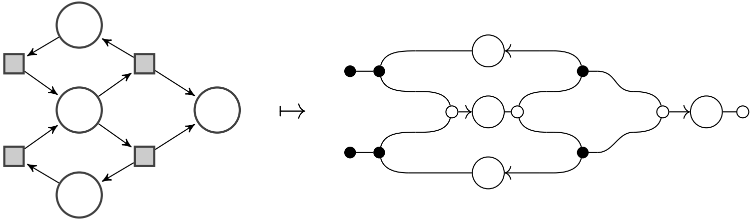
Embedding Petri nets



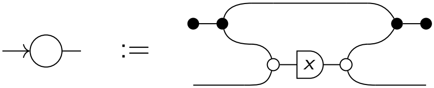
With new syntactic sugar:



Embedding Petri nets



With new syntactic sugar:



Theorem

Firing semantics of Petri nets = semantics of corresponding diagram

- We can use RC to (de)compose Petri nets and reason equationally about their behaviour.

Moral of the story

Seemingly diverse computational models can be studied within the same algebraic/categorical framework.

This is not just another process algebra, with pictures.

More coming: a graphical assembly language

- Affine extension (done): discrete *polyhedral* relations to capture non-additive phenomena like mutual exclusion.
- RC is strictly more expressive than Petri nets: compile other process algebras into it.
- Coarser semantics:
  - trace equivalence
  - bisimulation
- Compositional reachability checking.