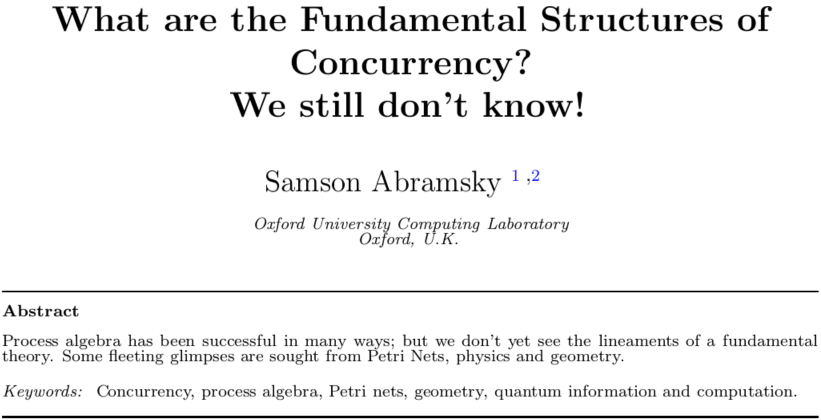
Picturing Resources in Concurrency: from Linear to Additive Relations

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

SYCO 2

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.

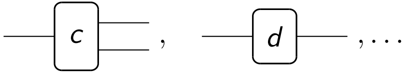
Process algebras vs. Petri nets

In this talk

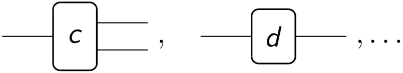
We try to bridge the gap between the two approaches.

- Start from a simple 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.

Drawing open systems



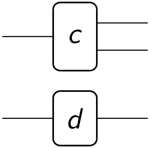
Drawing open systems



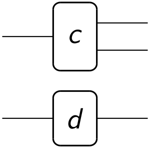
$$\mapsto$$

$[\![c]\!]_X \subseteq X \times X^2$, $[\![d]\!]_X \subseteq X \times X \dots$ for some fixed set X.

Parallel composition



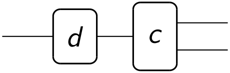
Parallel composition



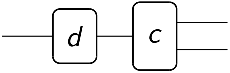
$$\downarrow$$

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

Synchronising composition



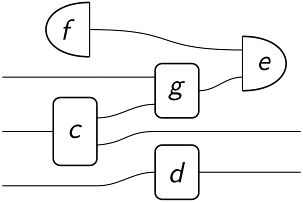
Synchronising composition



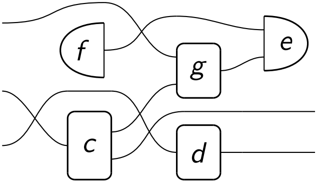
$$\mapsto$$

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

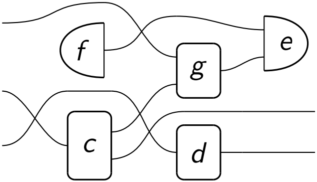
More complex networks



Only the connectivity matters

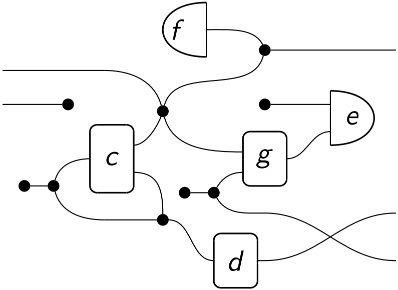


Only the connectivity matters

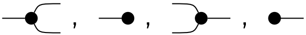


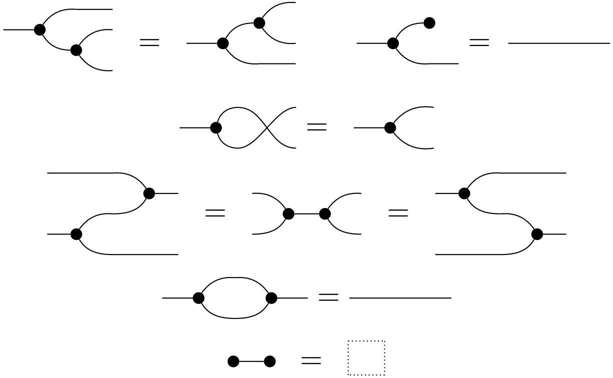
$$\left[\!\left[\, \times \,\right]\!\right]_X = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix} \right) \,\middle|\, x, y \in X \right\}$$

Multiple connections



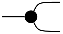
Frobenius monoids

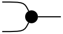
Special boxes/systems:  satisfying:



- form a *special commutative Frobenius monoid*.

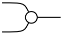
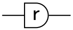
Interpreted as:

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

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

More algebraic structure

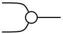
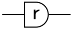
If $X = \mathrm{R}$ is a semiring we buy ourselves more structure:

, and 

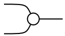
for $r \in R$

More algebraic structure

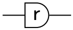
If $X = \mathsf{R}$ is a semiring we buy ourselves more structure:

, and  for $r \in R$

↧

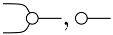
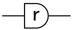
$[\![$  $]\!]_{\mathsf{R}} = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, x + y \right) \mid (x, y) \in \mathsf{R}^2 \right\}$

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

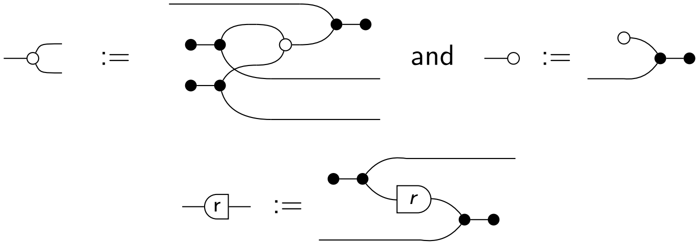
$[\![$  $]\!]_{\mathsf{R}} = \{(x, rx) \mid x \in \mathsf{R}\}$

More algebraic structure

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

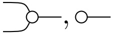
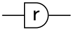
 and  for $r \in R$

and tranposes for free:



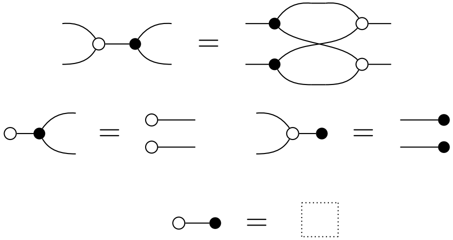
More algebraic structure

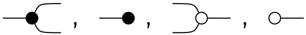
If $X = \mathrm{R}$ is a semiring we buy ourselves more structure:

 and 

for $r \in R$

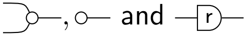
satisfying:



-  form a *bimonoid*.

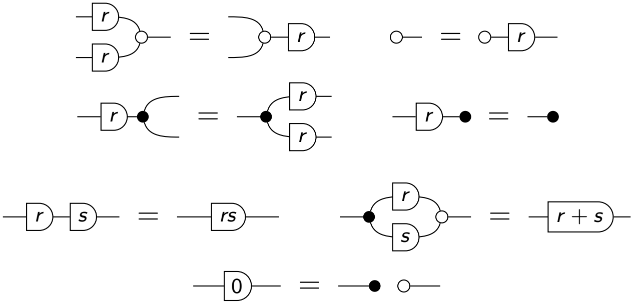
More algebraic structure

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



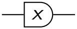
for $r \in R$

satisfying:



- Encode the additive and multiplicative operations of R.

Adding state

- Introduce  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

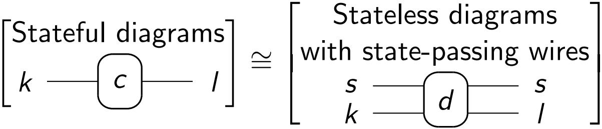
  $$[\![\, c \,]\!] \subseteq \mathsf{R}^{s+k} \times \mathsf{R}^{s+l}$$

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

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

The register is canonical

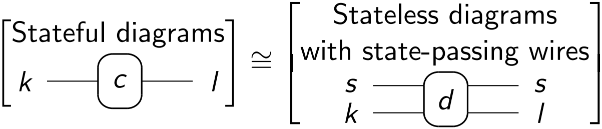
Isomorphism of props (*if* they are traced monoidal):



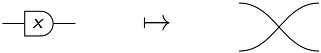
Stateful diagrams ≅ Stateless diagrams with state-passing wires

The register is canonical

Isomorphism of props (*if* they are traced monoidal):

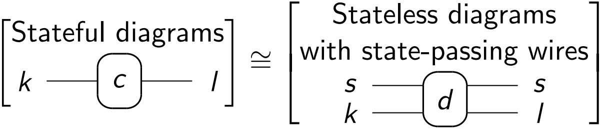


[Stateful diagrams] ≅ [Stateless diagrams with state-passing wires]

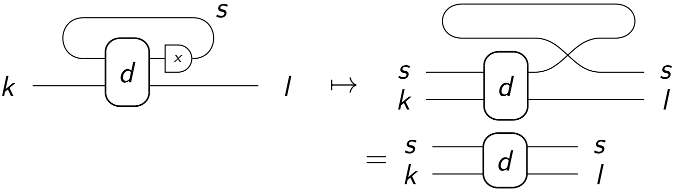


The register is canonical

Isomorphism of props (*if* they are traced monoidal):



Stateful diagrams ≅ Stateless diagrams with state-passing wires

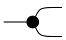


Section 2

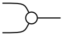
The Linear Interpretation

The prop of linear relations

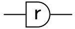
As relations over a field $\mathbb{K}$:

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

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

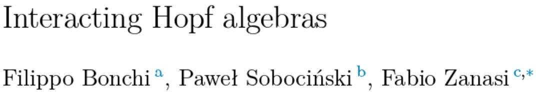
$[\![$  $]\!]_{\mathbb{K}} = \left\{ \left(\begin{pmatrix} x \\ y \end{pmatrix}, x + y \right) \mid (x, y) \in \mathbb{K}^2 \right\}$

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

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

- 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

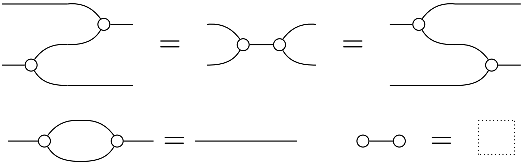


Interacting Hopf algebras

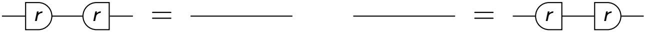
Filippo Bonchi[a], Paweł Sobociński[b], Fabio Zanasi[c,*]

JOURNAL OF PURE AND APPLIED ALGEBRA

- Addition is also a special commutative Frobenius monoid:



- Scalars are invertible:

 for $r \neq 0$

Linear dynamical systems

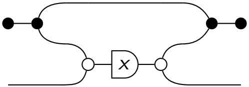
For the stateful linear case:

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

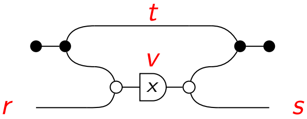
Section 3

The Resource Interpretation

Motivating example

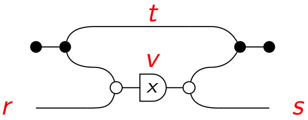


Motivating example



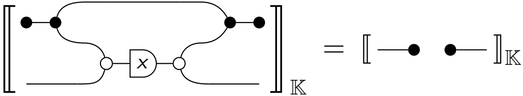
for $v = t + s$

Motivating example

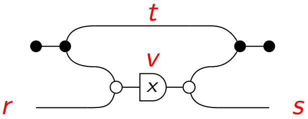


for $v = t + s$

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

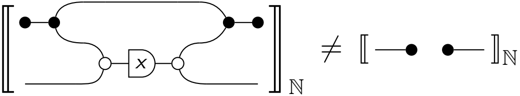


Motivating example

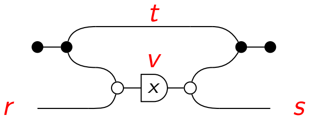


for $v = t + s$

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

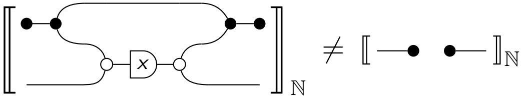


Motivating example



for $v = t + s$

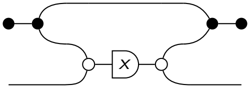
Over $\mathbb{N}$, we must have $s \leq v$ so:



Intuition

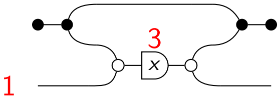
Without additive inverses, we cannot borrow arbitrary quantities.

Motivating example

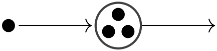


Over $\mathbb{N}$, this diagram behaves like the *place of a Petri net!*

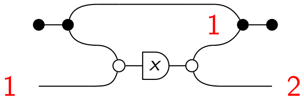
Motivating example



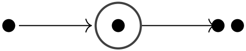
Over $\mathbb{N}$, this diagram behaves like the *place of a Petri net!*



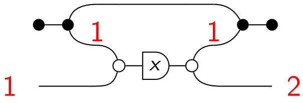
Motivating example



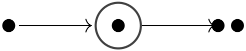
Over $\mathbb{N}$, this diagram behaves like the *place of a Petri net!*



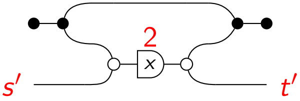
Motivating example



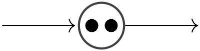
Over $\mathbb{N}$, this diagram behaves like the *place of a Petri net!*



Motivating example

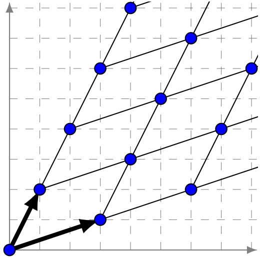


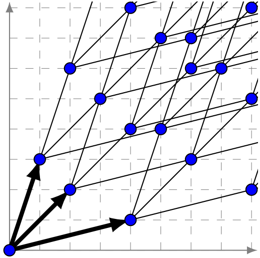
Over $\mathbb{N}$, this diagram behaves like the *place of a Petri net!*



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 $(\mathbf{0}, \mathbf{0})$.





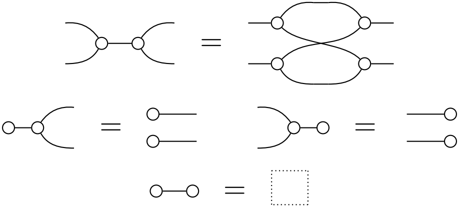
Proposition

Finitely-generated additive relations form a prop, AddRel*.*

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

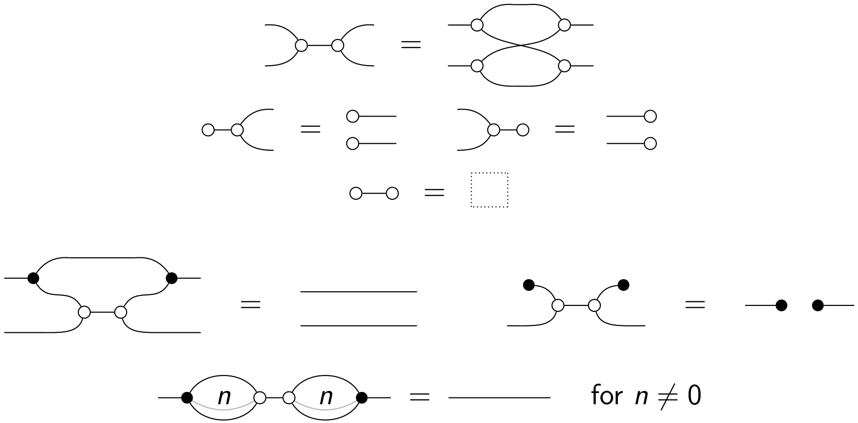
Complete equational theory

The Resource Calculus (RC) $\cong$ AddRel

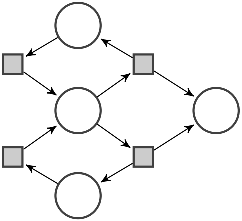


Complete equational theory

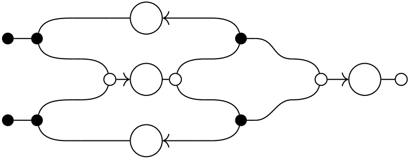
The Resource Calculus (RC) $\cong$ AddRel



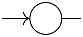
Embedding Petri nets



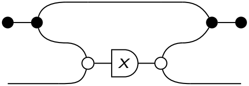
$\mapsto$



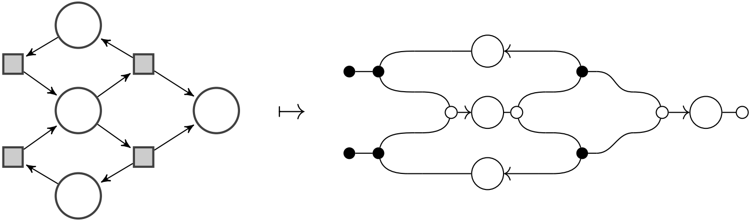
With new syntactic sugar:



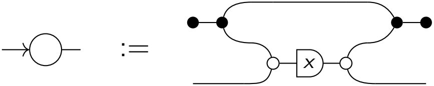
:=



Embedding Petri nets



With new syntactic sugar:



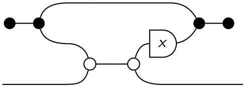
Theorem

Firing semantics of Petri nets = semantics of corresponding diagram

- We can use RC to reason equationally about Petri nets.

An assembly language

We can change the usual operational semantics of Petri nets using RC. Consider, e.g,



$$\mapsto \left\{ \left(\begin{pmatrix} n \\ i \end{pmatrix}, \begin{pmatrix} m \\ o \end{pmatrix} \right) \mid i + n = m + o \right\}$$

- Banking semantics

In summary

- Started from a generic diagrammatic language.
- Provided a resource-conscious semantics to model concurrent phenomena, e.g., Petri nets.
- Axiomatised this interpretation by giving a sound and complete equational theory.
- Seemingly diverse computational models can be studied within the same algebraic/categorical framework.

Much more to be done

- Affine extension (done): discrete polyhedral relations to capture mutual exclusion and more.
- Coarser semantics:
  - streams for trace equivalence,
  - more behavioural equivalence, like bisimulation.
- Compositional reachability checking.
- Beyond Petri nets: compile process algebras into RC.