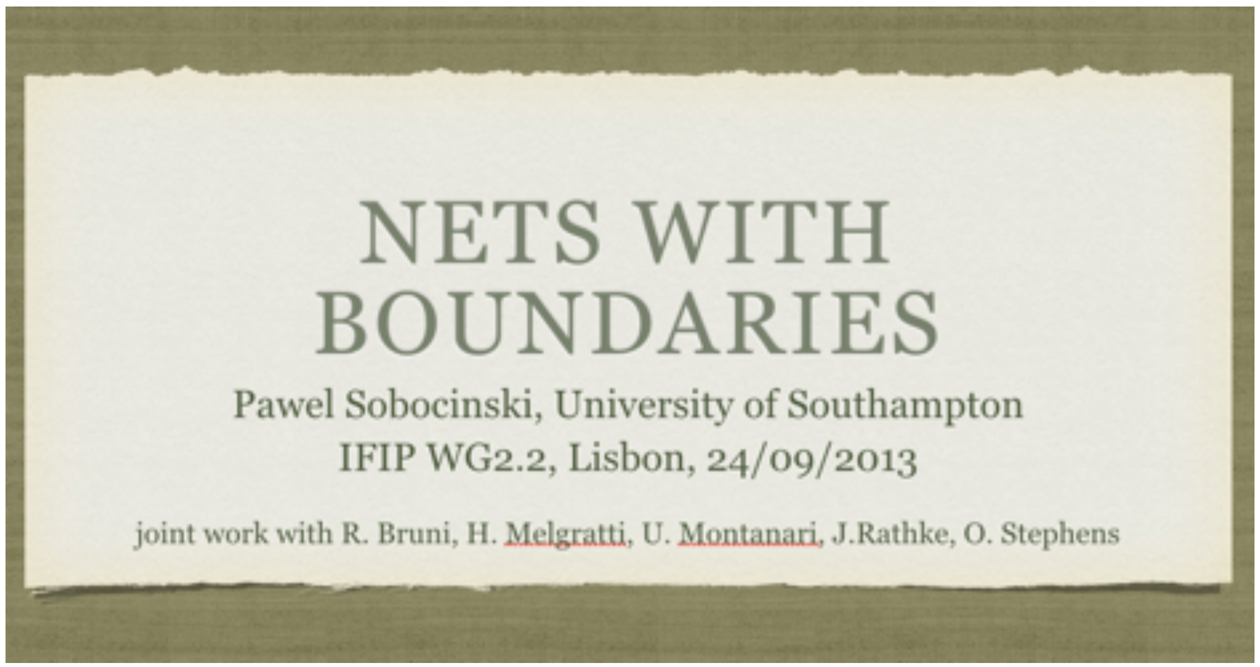
Control theory meets operational semantics

Pawel Sobocinski, University of Southampton

IFIP WG2.2 Lucca

My previous WG2.2 talks



Lisbon

Munich

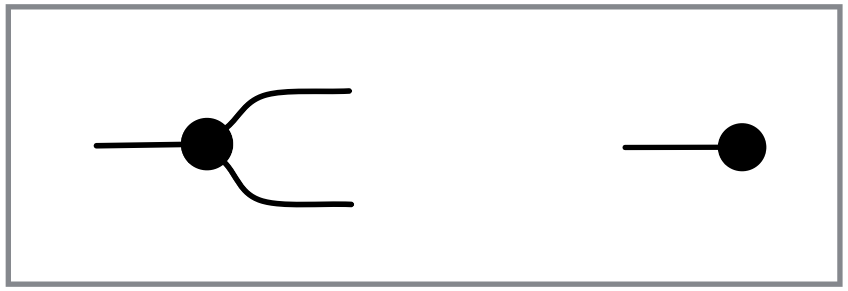
Today: What’s the big picture?

Plan

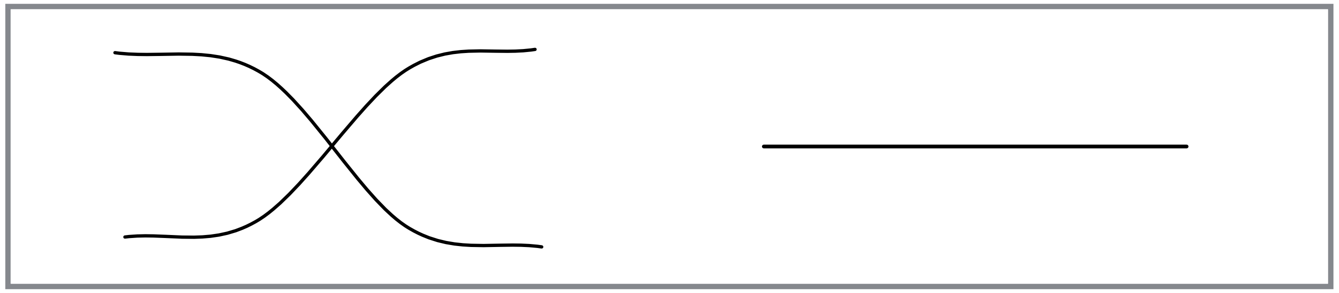
- **Symmetric monoidal theories**
- Applications
  - Petri nets (joint work with Owen Stephens and Julian Rathke)
  - Signal flow graphs (joint work with Filippo Bonchi and Fabio Zanasi)
- Ongoing research
  - Control theory (joint work with Brendan Fong and Paolo Rapisarda)
  - Graph theory (joint work with Apiwat Chantawibul)
  - Linear algebra (personal hobby)

Symmetric monoidal syntax

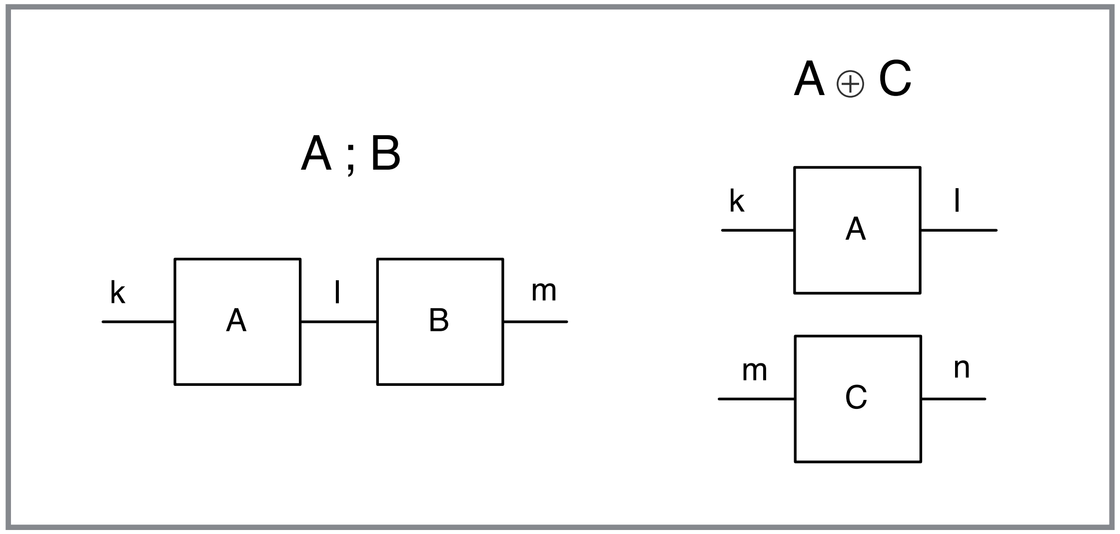
- generators (e.g.)



- basic tiles

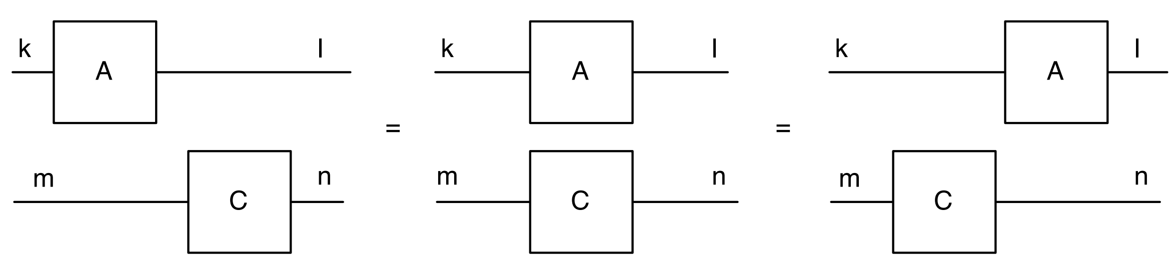


- algebra

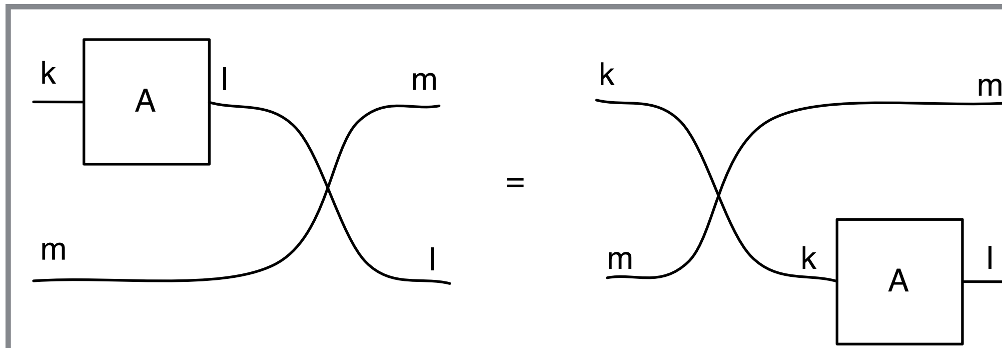


Term equality

- diagrams can slide along wires

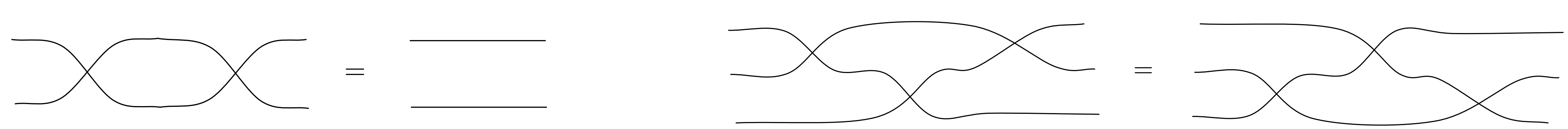


functoriality



naturality

- wires don't tangle, i.e.



i.e. pure wiring obeys the same equations as permutations

- sub-diagrams can be replaced with equal diagrams (compositionality)

Equations

- call a SM syntax + *equations* a **symmetric monoidal theory (SMT)**

- any SMT is a special kind of a monoidal category called PROP (product and permutation category)

- mathematical structures often organise themselves as arrows of a PROP **C** — finding an SMT characterisation **T** (i.e. **T** ≅ **C**) is thus a fully complete axiomatisation

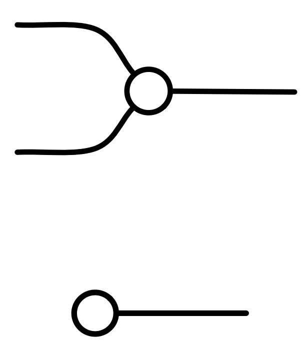
- for a sound and complete axiomatisation, it suffices to find a faithful homomorphism **T** → **C**

PROPs

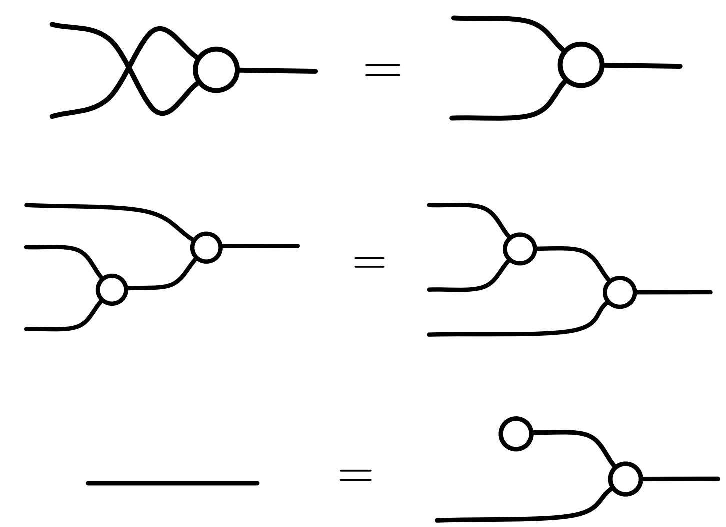
- (product and permutation categories)
  - strict symmetric monoidal (monoidal product is associative on the nose)
  - objects = natural numbers
  - monoidal product on objects = addition
- e.g. the PROP **F** where arrows from m to n are the functions from [m] = {0,1,…, m-1} to [n]

Example: Functions, diagrammatically

Generators

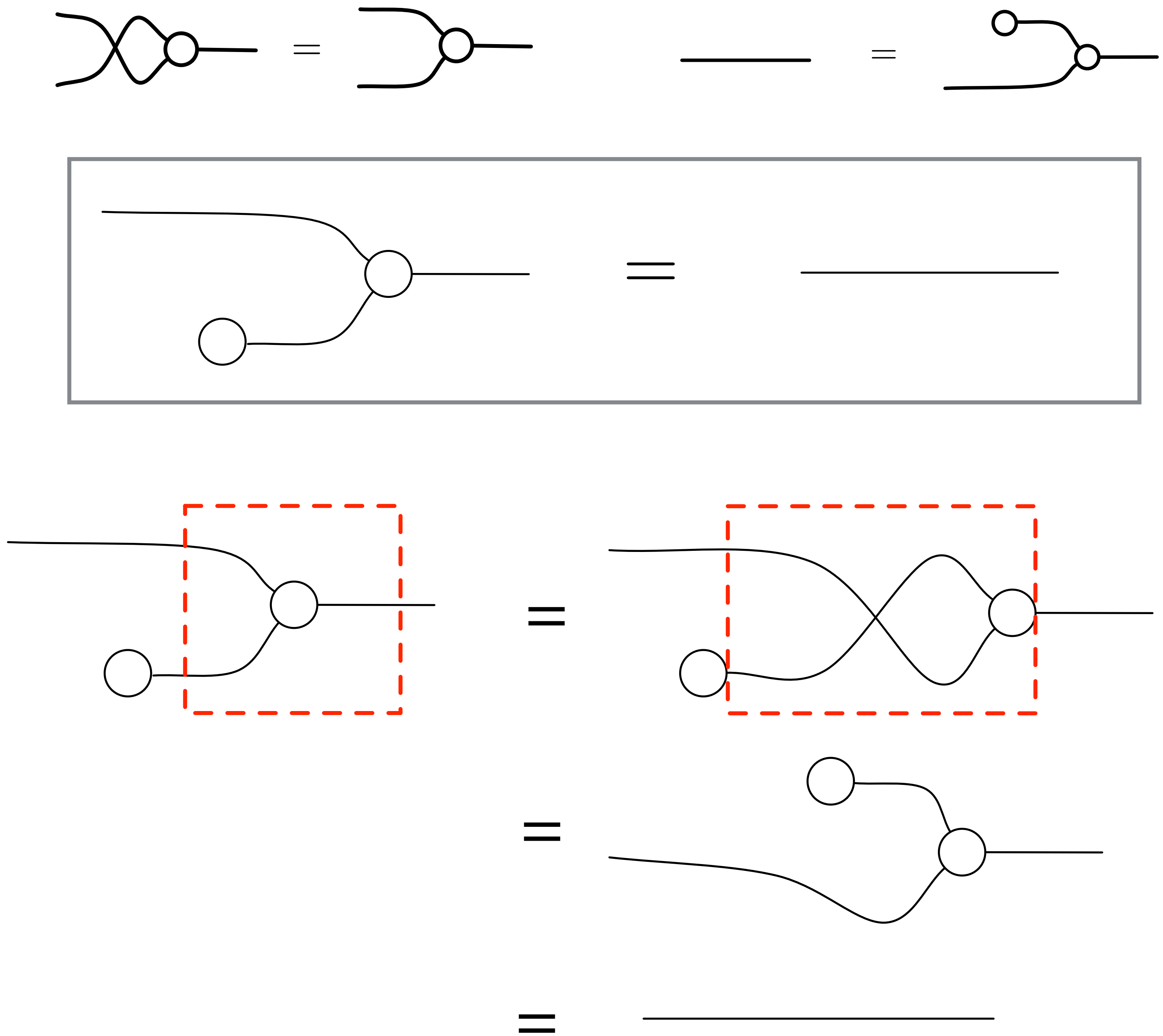


Equations



- SMT **M** on this data isomorphic to the PROP **F** of functions
- i.e. the "commutative monoids are the SMT of functions"

Diagrammatic reasoning example

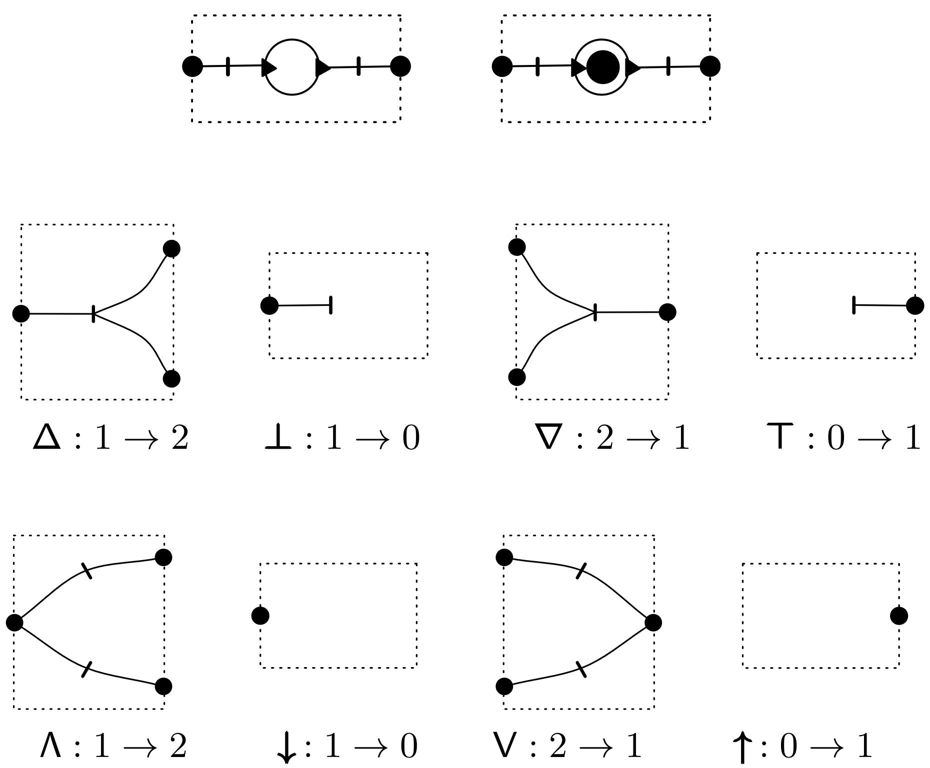


Plan

- Symmetric monoidal theories
- **Applications**
  - Petri nets (joint work with Owen Stephens and Julian Rathke)
  - Signal flow graphs (joint work with Filippo Bonchi and Fabio Zanasi)
- Ongoing research
  - Control theory (joint work with Brendan Fong and Paolo Rapisarda)
  - Graph theory (joint work with Apiwat Chantawibul)
  - Linear algebra (personal hobby)

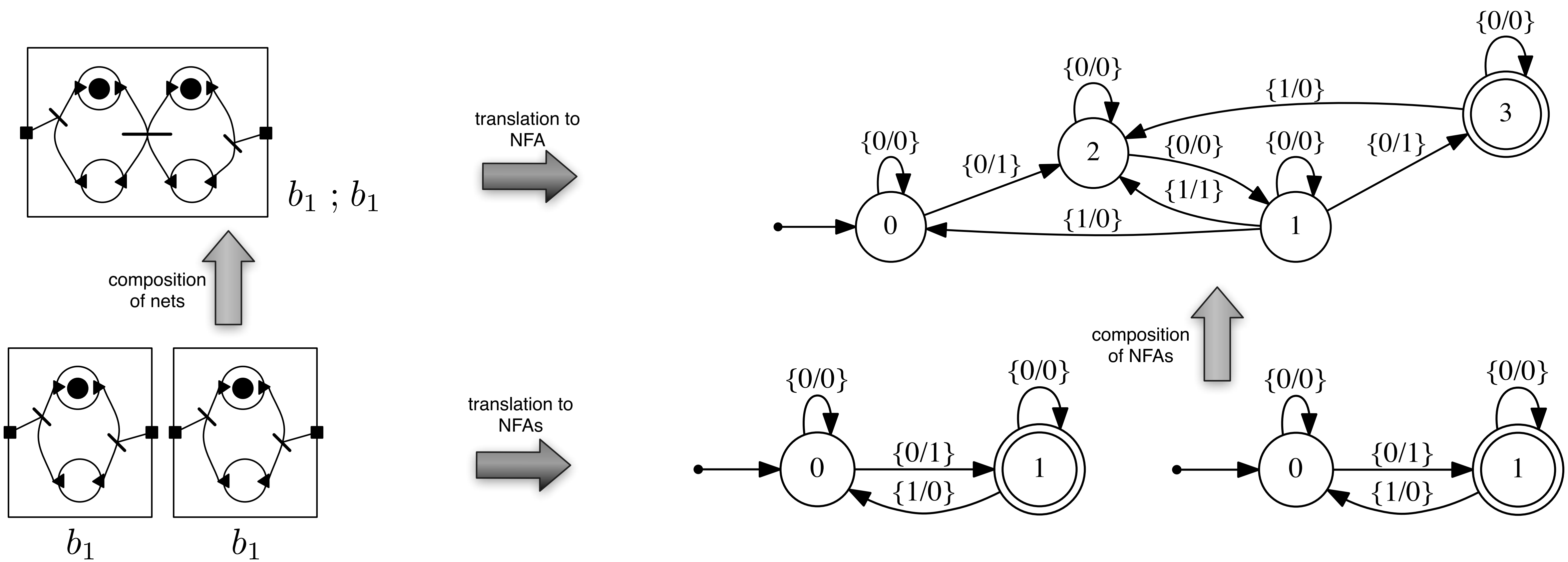
Petri nets

(CONCUR `10, `11, Petri Nets `14, Reachability Problems `14)



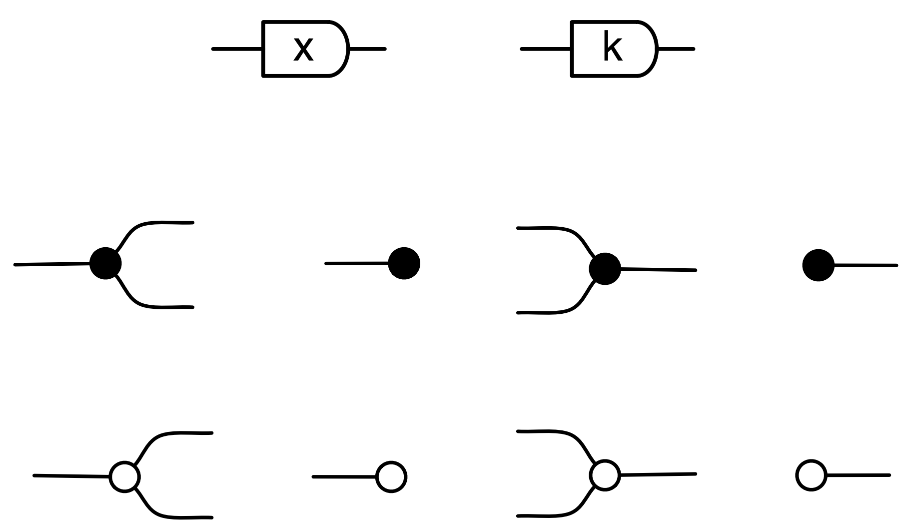
- all nets can be constructed from these generators
- compositional SOS semantics in terms of 2-labelled transition systems
    - 1-safe variant and P/T variant
- in suitable examples, compositionality can be used to (vastly) improve efficiency of model checking
- enables parametric model checking

Example


$b_1 ; b_1$
composition of nets
b_1
b_1
translation to NFA
translation to NFAs
composition of NFAs
{0/0}
{0/1}
{1/0}
{1/1}
0
1
2
3

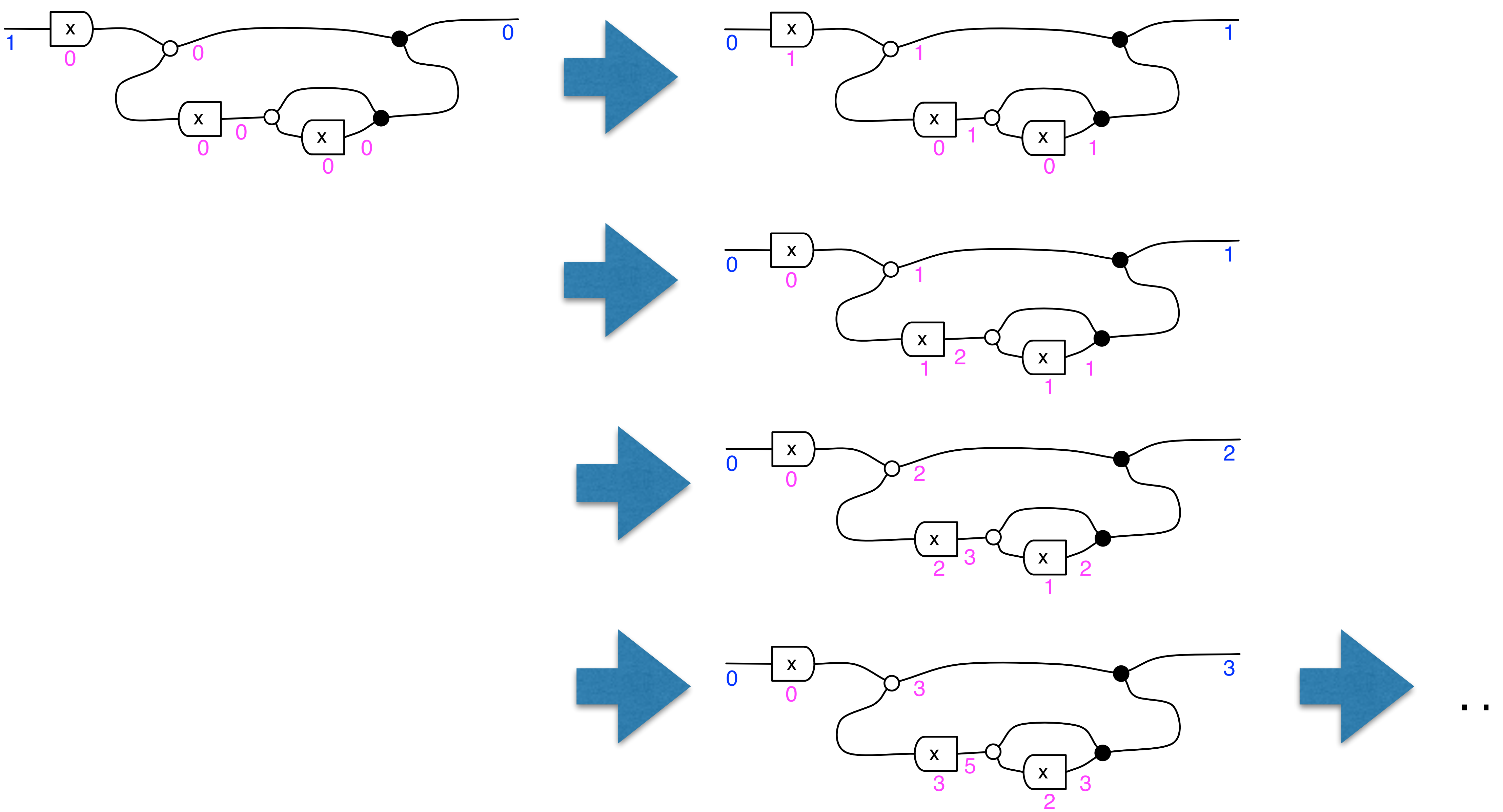
Signal flow graphs

(FoSSaCS `13, CONCUR `14, PoPL `15)



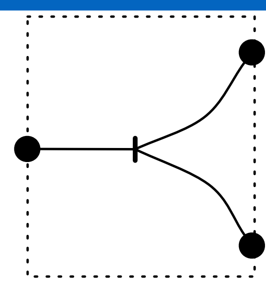
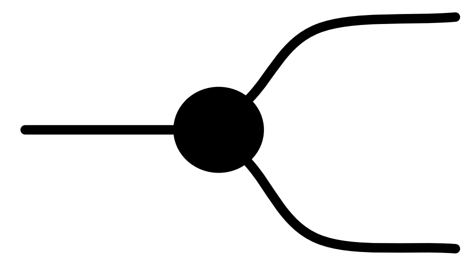
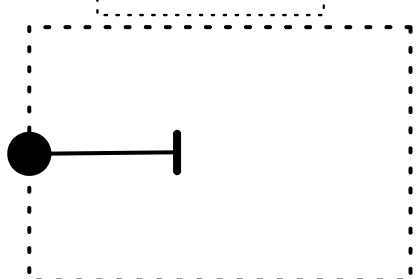
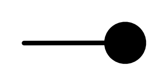
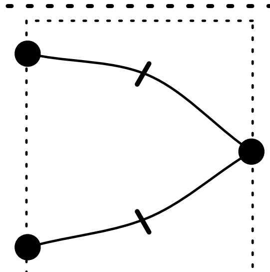
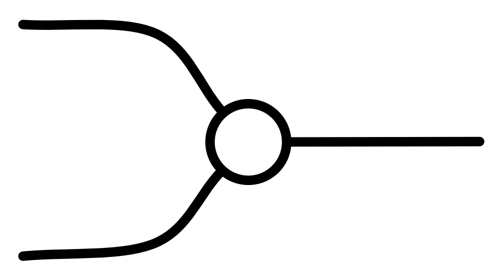
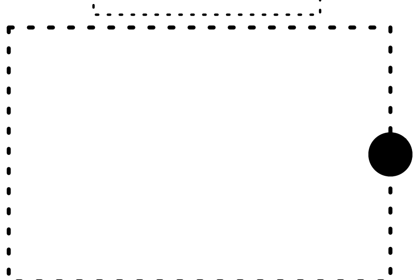
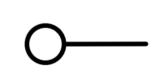
- all signal flow graphs can be constructed from these generators
- compositional operational semantics in terms of 2-labelled transition systems
- semantics is *executable* when the direction of signal flow is consistent
- captures a canonical class of linear time-invariant dynamical systems

Example

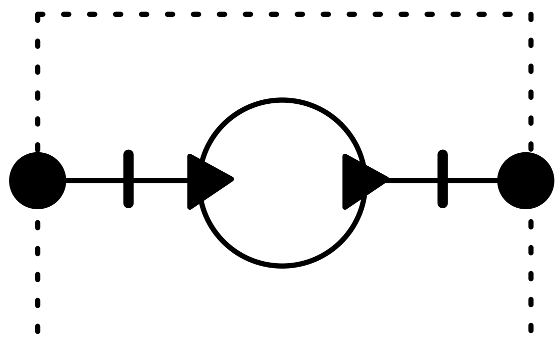
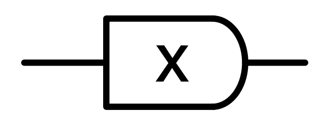
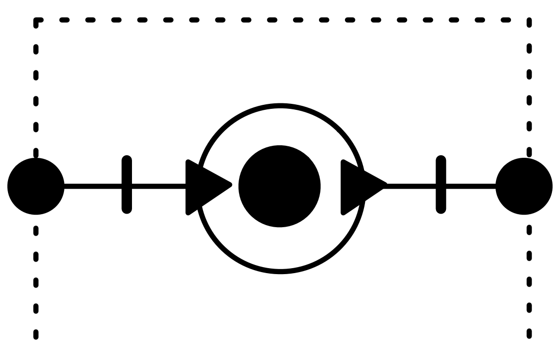
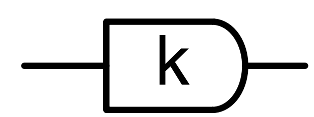


x
1
0
0
0
x
0
0
x
0
0
x
0
1
1
1
x
0
1
x
0
1
x
0
1
0
1
x
1
2
x
1
1
x
0
2
0
2
x
2
3
x
1
2
x
0
3
0
3
x
3
5
x
2
3
...

Connectivity

Petri Nets		Signal Flow Graphs	
	transition can connect to multiple places		copy signal
	no further connections		discard signal
	place can connect to multiple transitions		add signals
	no transition		emit zero

Other generators

Petri Nets (safe nets)		Signal Flow Graphs	
	empty place		one space buffer
	place with token		amplify signal (mutiply) by k

Other applications of symmetric monoidal syntax

- Categorical Quantum Information - Abramsky & Coecke `04, Coecke & Duncan `08
- Modelling asynchronous circuits - Ghica `14
- Directed acyclic graphs - Fiore & Campos `13
- …

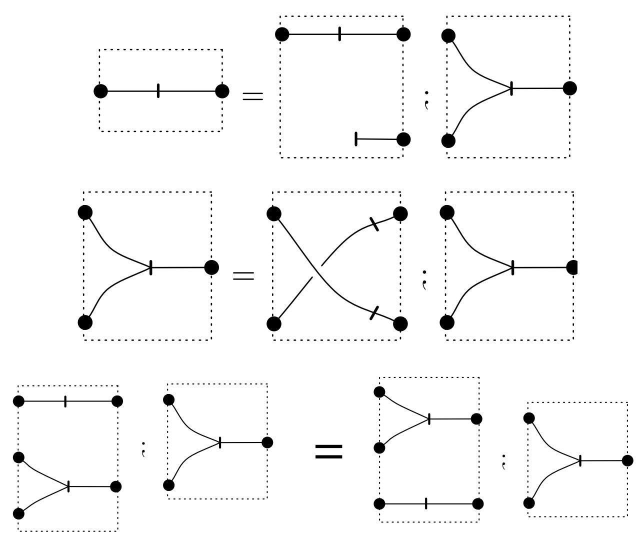
Capturing semantic equivalence

- Once we know a syntax, and the intended semantics, can we characterise semantic equivalences equationally?

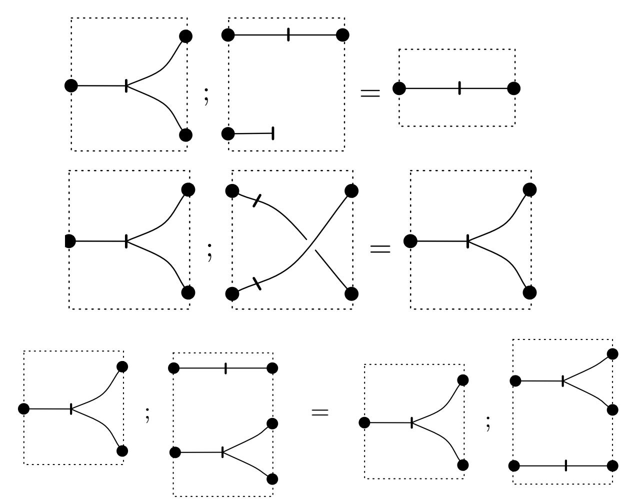
- In Petri nets and signal flow graphs, doing so identifies several interesting algebraic structures

Example 1: Petri nets

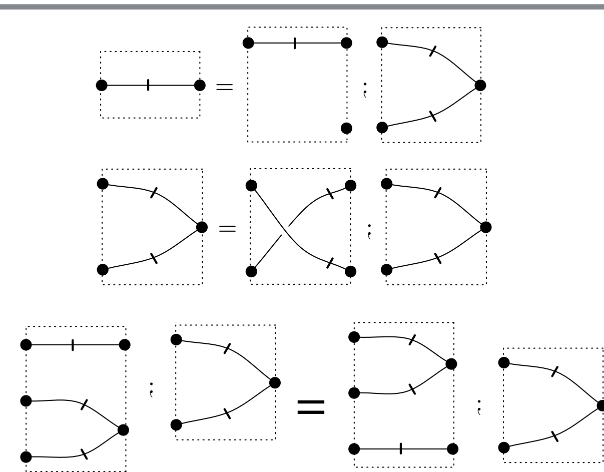
(CALCO `13)



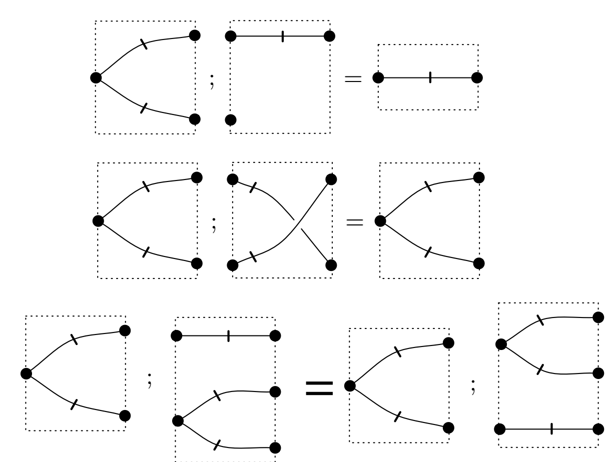
commutative monoid



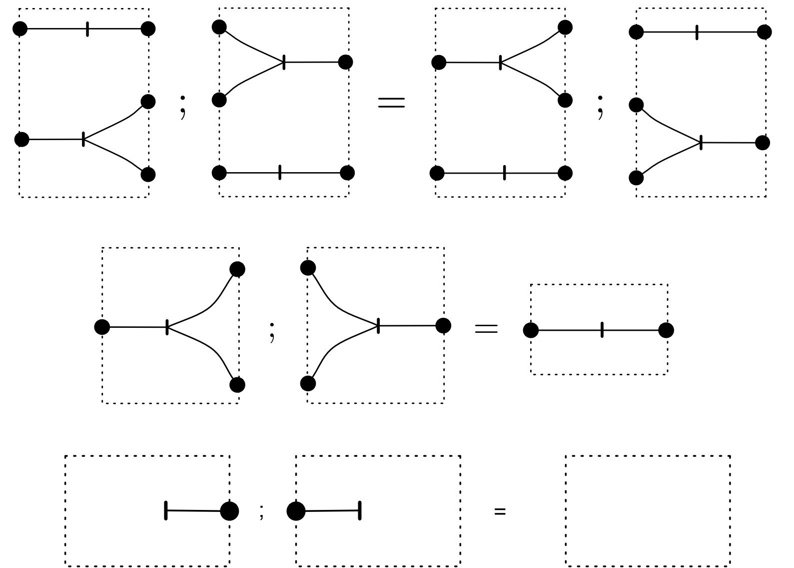
commutative comonoid



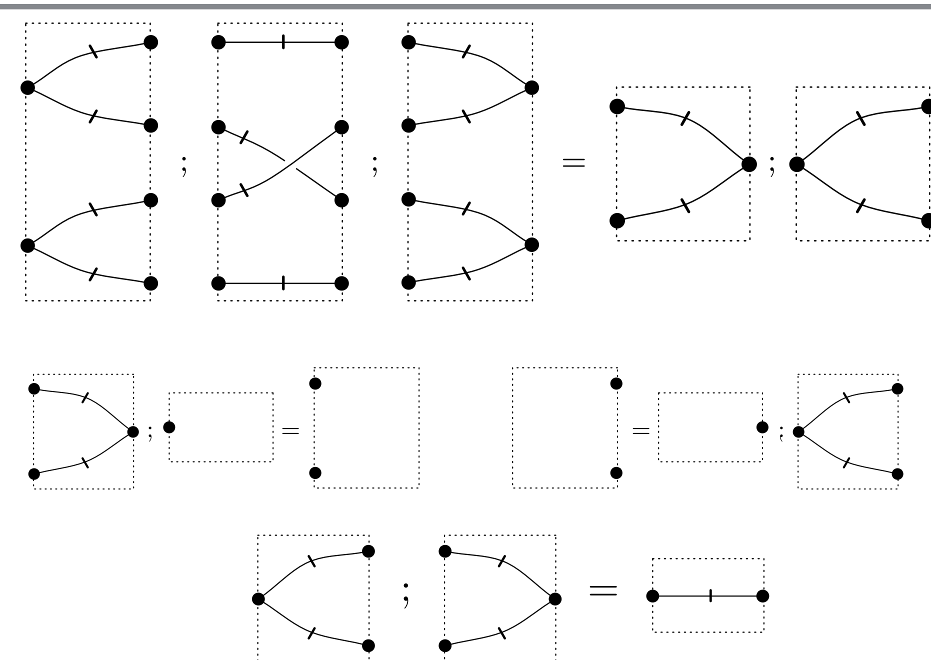
commutative monoid



commutative comonoid



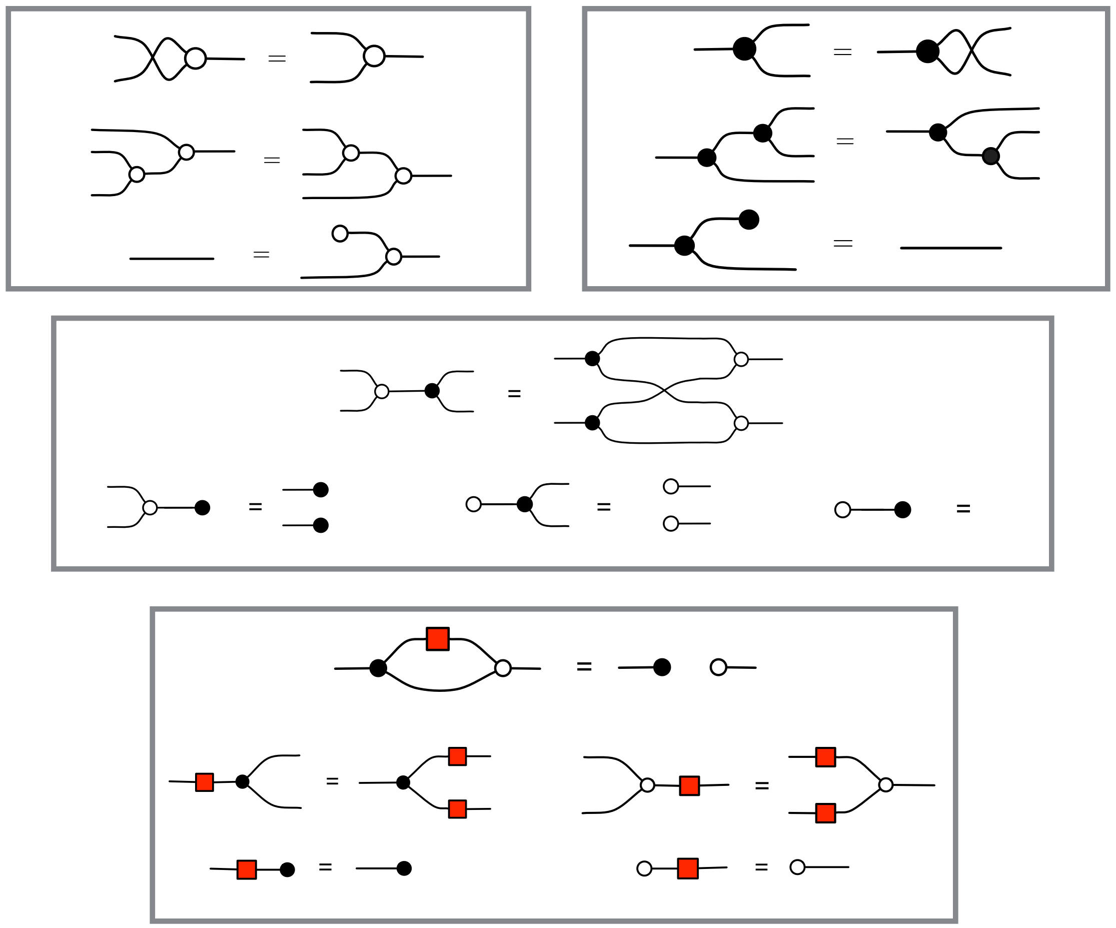
special Frobenius (bi)monoid



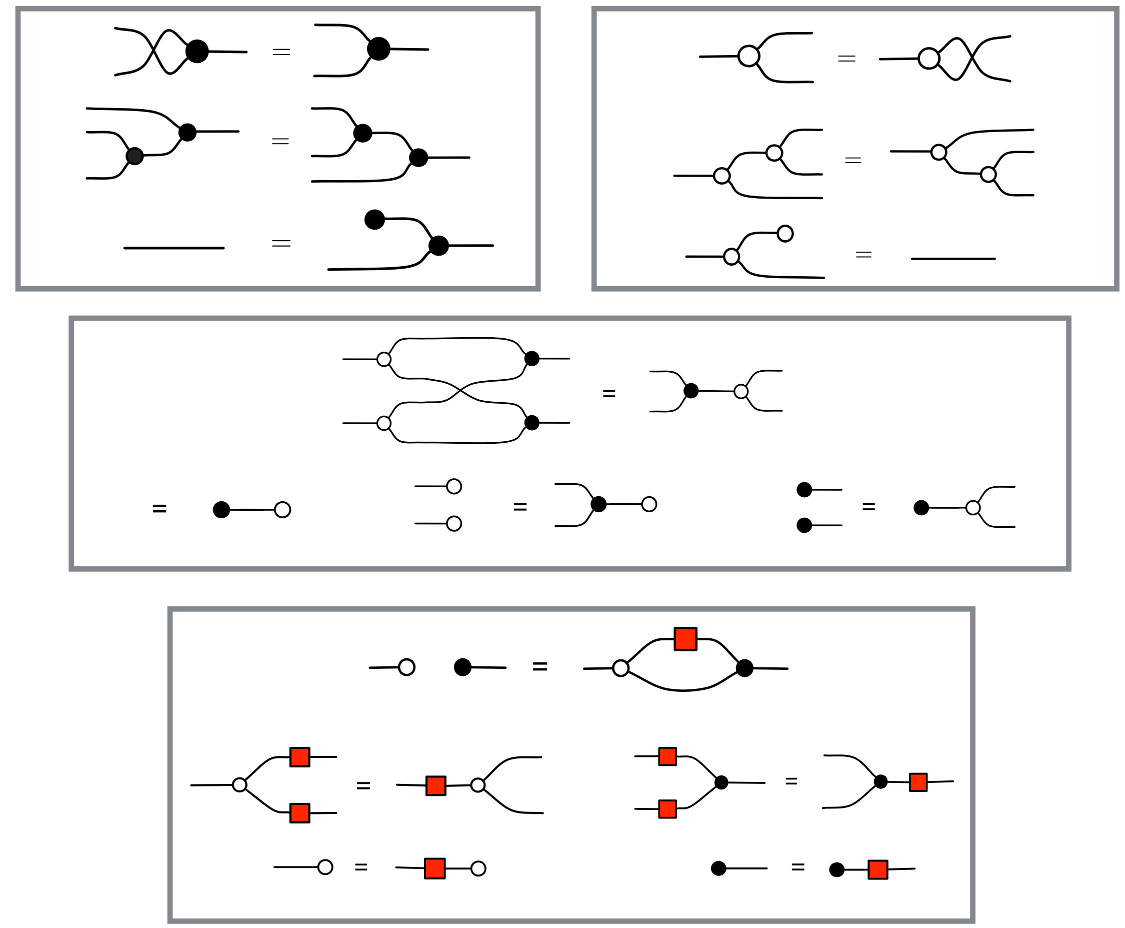
special bimonoid (infinite state case)

other equations become a bit more involved

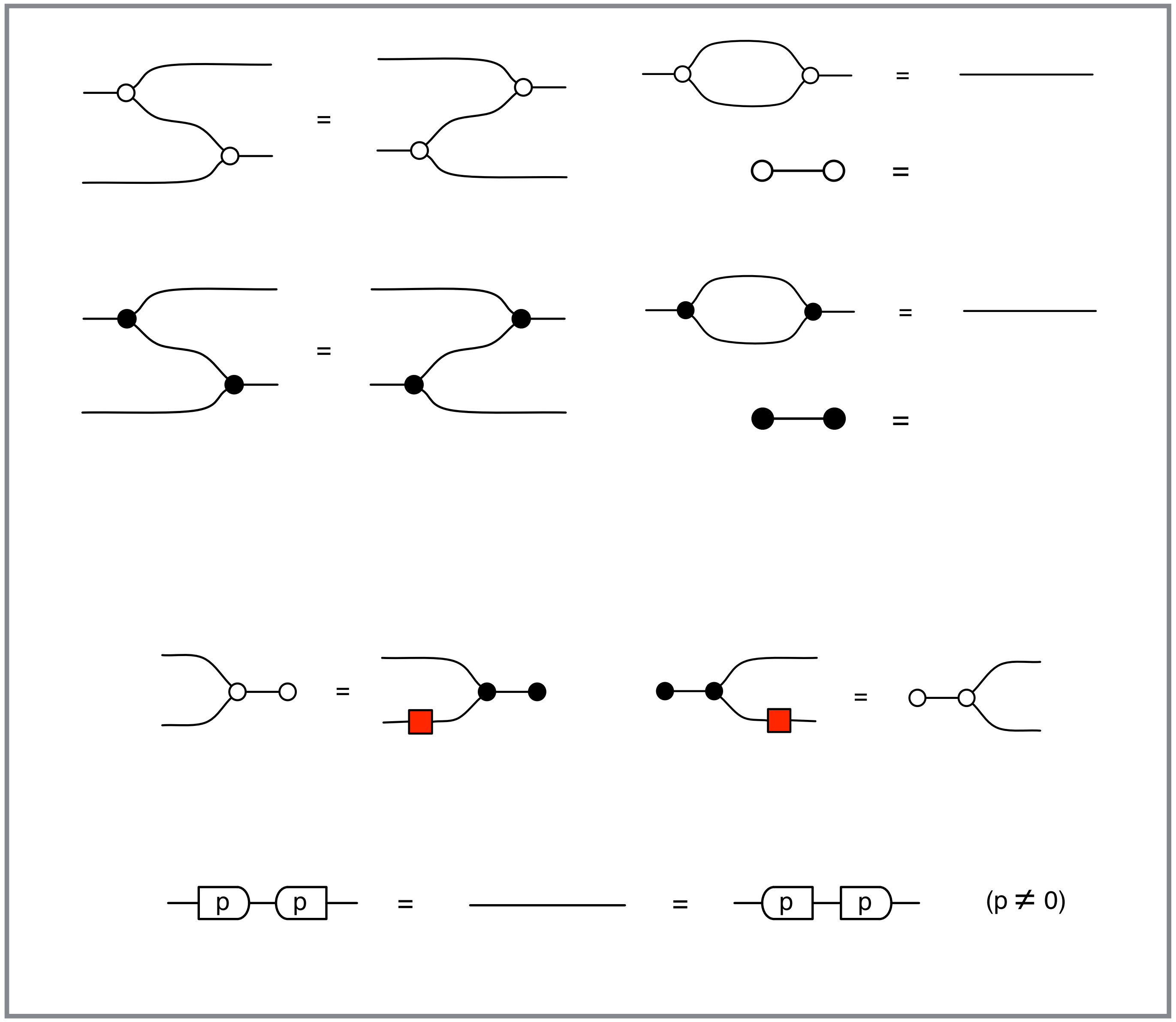
Example 2: Signal flow graphs



Hopf algebra



Hopf algebra



Interacting Hopf monoids

- For any principal ideal domain (PID) R, the SMT $\mathbf{IH}_{R}$ of interacting Hopf Monoids characterises the PROP $\mathbf{LinRel}_{ff(R)}$
- example on the left is for the PID of integers, and gives the equational characterisation of $\mathbf{LinRel}_{\mathbf{Q}}$

$\mathbf{LinRel}_{\mathbf{Q}}$

- PROP of linear relations over the rationals
  - arrows m to n are linear subspaces of $\mathbf{Q}^m \times \mathbf{Q}^n$
  - composed **as relations**
  - monoidal product is direct sum
- **IH** is isomorphic to **LinRel**

SMT for signal flow: $\mathbf{IH}_{k[x]}$

- Isomorphism between $\mathbf{IH}_{k[x]}$ and $\mathbf{LinRel}_{k(x)}$
- The field of fractions of k[[x]] is the field of Laurent series k((x))
  - streams infinite in the future, but finite in the past
- There is a faithful homomorphism $\mathbf{LinRel}_{k(x)} \to \mathbf{LinRel}_{k((x))}$
- So the graphical calculus is a sound and complete language for linear relations over Laurent series

Full abstraction and realisability

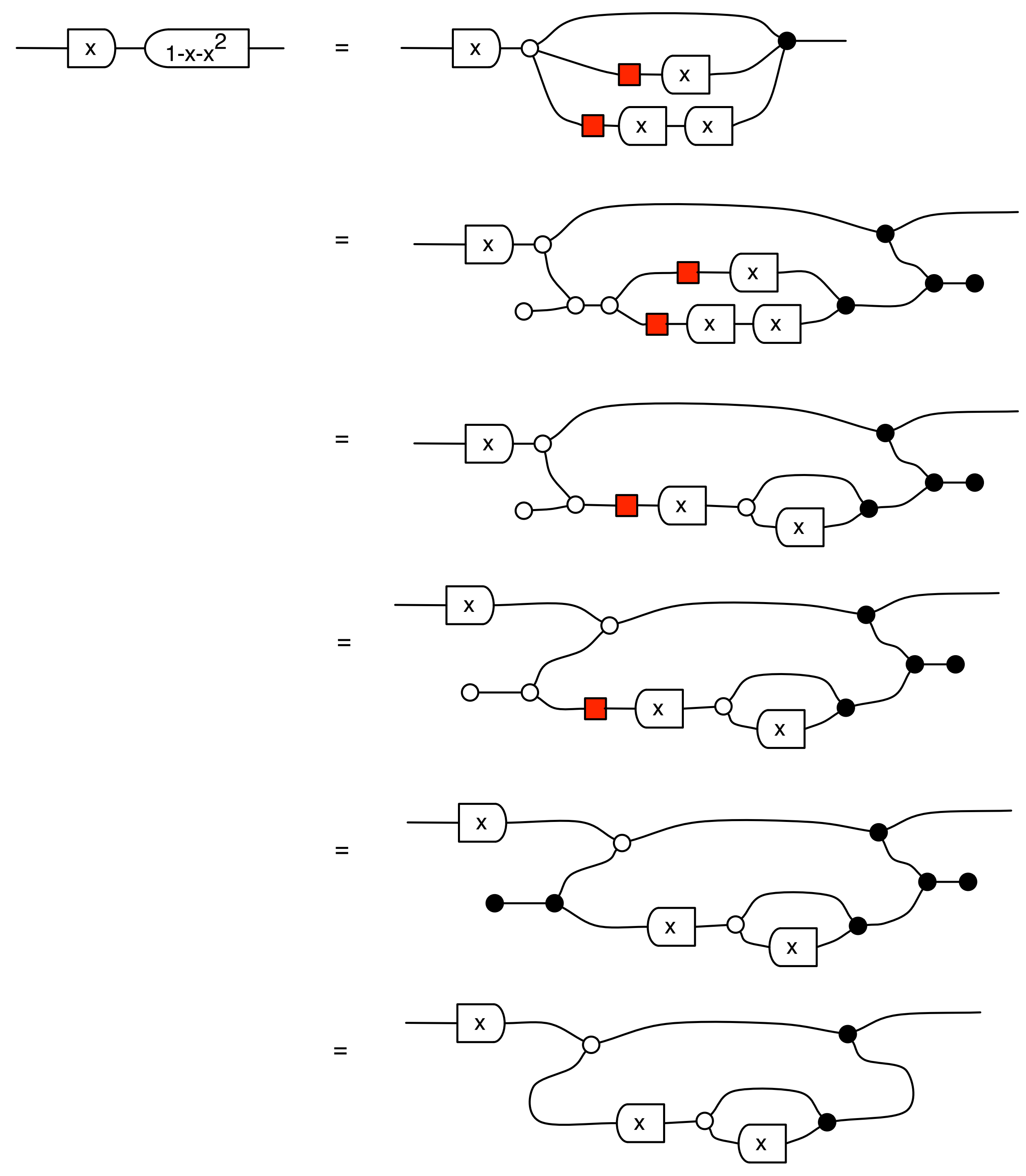
Bonchi, S, Zanasi, PoPL`15

- Full Abstraction
  - in any two circuits where the direction of flow is consistent, they have the same operational semantics iff they can be shown equal equationally
- Realisability
  - any circuit can be rewritten (in at least one way) so that the direction of flow is consistent

Example - implementing Fibonacci

$$\frac{x}{1-x-x^2}$$

is the generating function for the Fibonacci sequence



Plan

- Symmetric monoidal theories
- Applications
  - Petri nets (joint work with Owen Stephens and Julian Rathke)
  - Signal flow graphs (joint work with Filippo Bonchi and Fabio Zanasi)
- **Ongoing research**
  - Control theory (joint work with Brendan Fong and Paolo Rapisarda)
  - Graph theory (joint work with Apiwat Chantawibul)
  - Linear algebra (personal hobby)

Ongoing research

- **using the diagrammatic language in control theory (with B. Fong and P. Rapisarda)**
- connections with graph theory (with A. Chantawibul)
- graphical linear algebra (personal hobby)

Signal flow graphs and controllability

- In work so far, the denation of a circuit is a linear relation over the field of Laurent series
  - streams infinite in the future and finite in the past
- Every behaviour is **controllable**

Signal flow with biinfinite stream semantics

- extending to streams infinite in both directions ($k^{\mathbb{Z}}$) adds uncontrollable behaviours
- a canonical subtheory of $\mathbf{IH}_{k[x]}$ gives us a sound and complete equational characterisation for $\mathbf{LinRel}_{k^{\mathbb{Z}}}$

Ongoing research

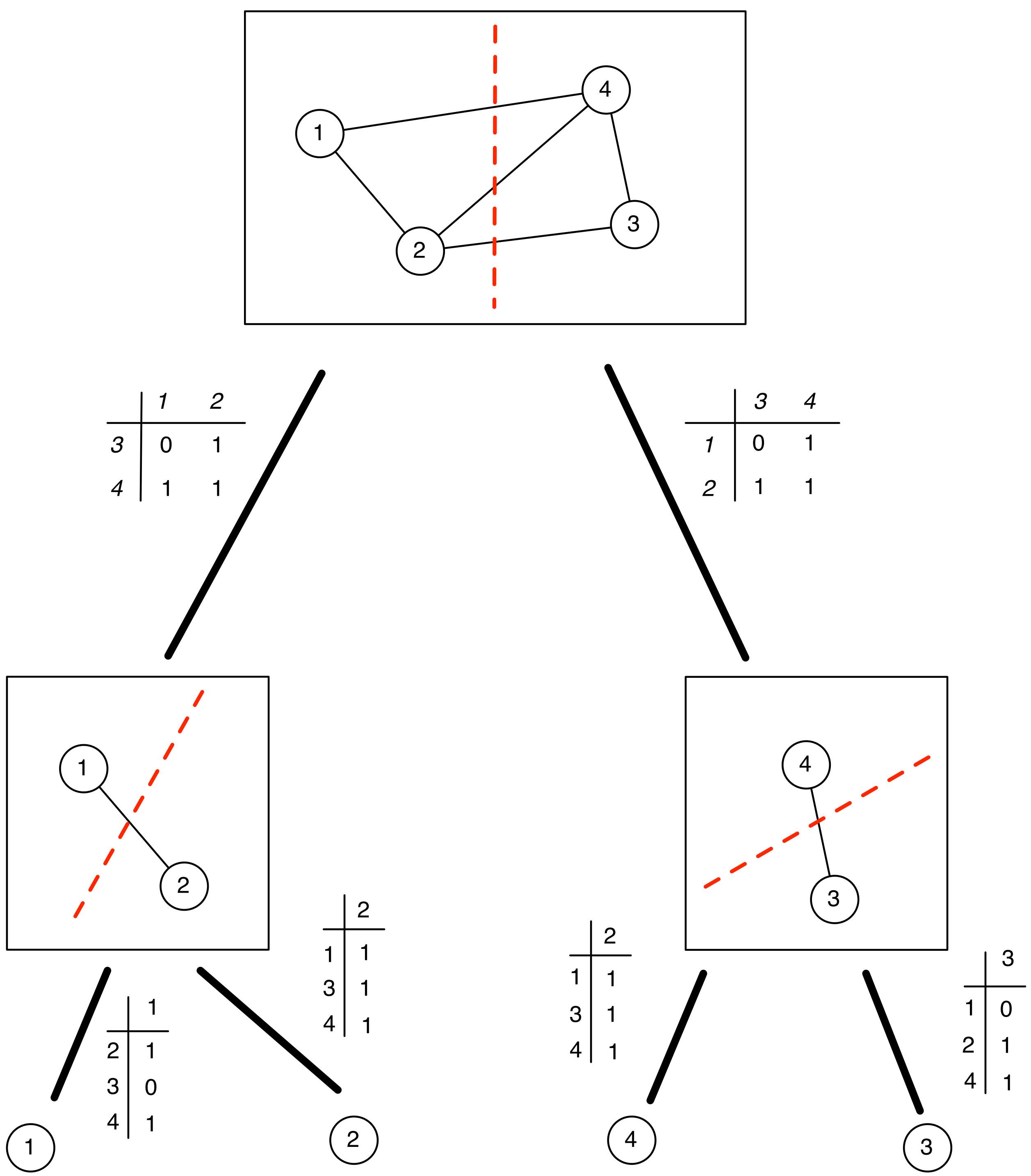
- using the diagrammatic language in control theory (with B. Fong and P. Rapisarda)
- **connections with graph theory (with A. Chantawibul)**
- graphical linear algebra (personal hobby)

Motivation

- often useful to know whether a string diagram can be decomposed in a “good” way
  - e.g. in a Petri net (Reachability Problems ’14) to enable divide-and-conquer compositional model checking
- graph theory gives us ideas for nice ways of decomposing graphs, as well as powerful metrics that measure how “difficult” a graph is
  - e.g. **structural metrics** such as path width, tree width, clique width, rank width, …

Example: rank decomposition

for simple graphs



rank of a decomposition
= maximal matrix rank

rank width = rank of optimal decomposition

(minmax/infsup flavour)

Can we understand this as an algebraic expression?

Decompositions in an SMT

(Chantawibul, S., MFPS`15)

- An SMT for open simple graphs
- Rank decompositions can be written as expressions in the SMT
- Relies on a technical notion, an extended notion of matrix called U-matrix
- Working on general theory of decompositions for SMTs, with rank decomposition of a simple graph as special case

Ongoing research

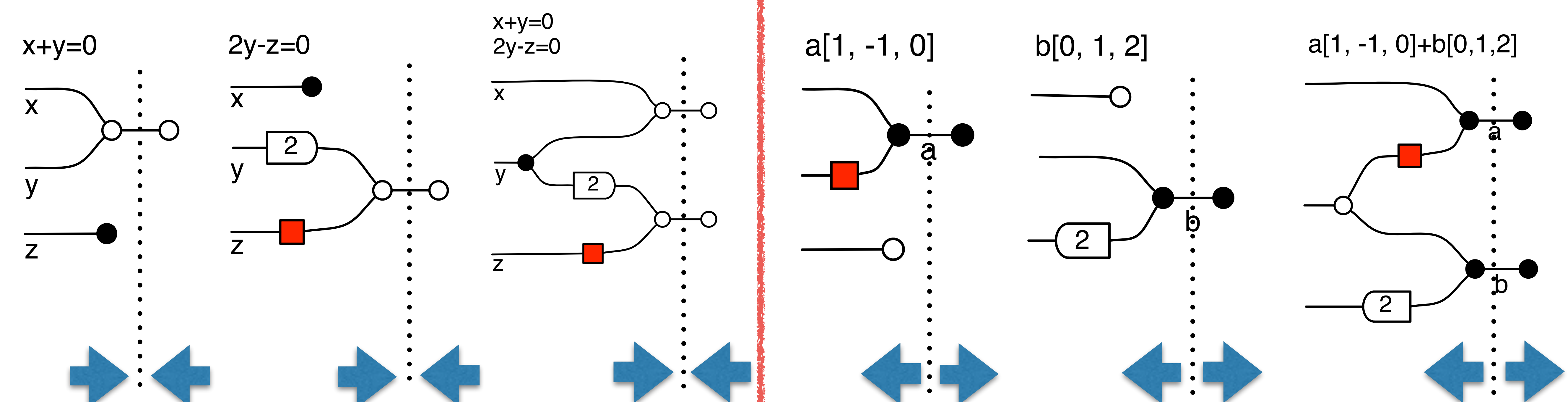
- using the diagrammatic language in control theory (with B. Fong and P. Rapisarda)
- connections with graph theory (with A. Chantawibul)
- **graphical linear algebra (personal hobby)**

Factorisations

- In the theory of interacting Hopf monoids, every diagram can be factorised as a span or a cospan of matrices
- This translates to the two different ways one can think of linear spaces

as solutions of homogeneous equations

as linear combinations of basis vectors



Cospans

Spans

Goal

- Develop a resource for linear algebra without the notions of
  - vector space as “set of vectors”
  - linear space as a set generated by basis elements
  - etc..
- See http://graphicallinearalgebra.net