Deconstructing the Calculus of Relations with Tape Diagrams

Filippo Bonchi[1], Alessandro Di Giorgio[1], Alessio Santamaria[1,2]

[1]University of Pisa

[2]University of Sussex

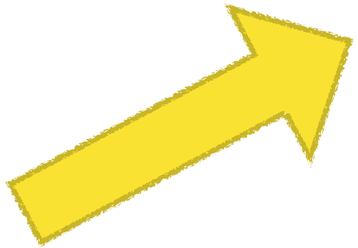
OPCT 2023 - Bertinoro - Italy

Motivations

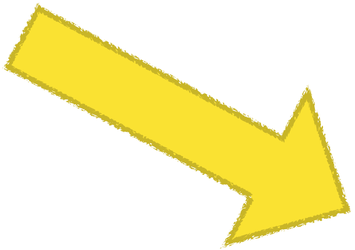
What are the fundamental structures of concurrency? We still don't know!

Samson Abramsky [1,2]

Oxford University Computing Laboratory
Oxford, U.K.



It starts with considerations similar to those by Hubert...



... but it ends up with rather different conclusions
(similar to those of Kim, Radu and Ilaria)

Motivations

What are the fundamental structures of concurrency? We still don't know!

Samson Abramsky [1,2]

Oxford University Computing Laboratory
Oxford, U.K.

It is too easy to cook up yet another variant process calculus or algebra; there are too few constraints. This *plasticity of definitions* has become so familiar in our field that we may not be aware of it as an issue. The mathematician André Weil apparently compared finding the right definitions in algebraic number theory — which was like carving adamantine rock — to making definitions in the theory of uniform spaces (which he founded), which was like sculpting with snow. In concurrency theory, we are very much at the snow-sculpture end of the spectrum. We lack the kind of external reality, whether it comes from fundamental mathematical objects like the integers, or manifolds, or differential equations, or from physical reality as determined by experiment, which is hard and obdurate, and resistant to our definitions. Is this a necessary feature of our existence, or have we just not yet found the real bedrock?

Motivations

What are the fundamental structures of concurrency? We still don't know!

Samson Abramsky [1,2]

Oxford University Computing Laboratory
Oxford, U.K.

It is too easy to cook up yet another variant process calculus or algebra; there are too few constraints. This *plasticity of definitions* has become so familiar in our field that we may not be aware of it as an issue. The mathematician André Weil apparently compared finding the right definitions in algebraic number theory — which was like carving adamantine rock — to making definitions in the theory of uniform spaces (which he founded), which was like sculpting with snow. In concurrency theory, we are very much at the snow-sculpture end of the spectrum. We lack the kind of external reality, whether it comes from fundamental mathematical objects like the integers, or manifolds, or differential equations, or from physical reality as determined by experiment, which is hard and obdurate, and resistant to our definitions. Is this a necessary feature of our existence, or have we just not yet found the real bedrock?

To design a new formal language is a bit like programming...

...and one can program with different languages

Programming in Javascript is rather easy, since the language poses few constraints, but then one can write horrible code

Motivations

What are the fundamental structures of concurrency? We still don't know!

Samson Abramsky [1,2]

Oxford University Computing Laboratory
Oxford, U.K.

It is too easy to cook up yet another variant process calculus or algebra; there are too few constraints. This *plasticity of definitions* has become so familiar in our field that we may not be aware of it as an issue. The mathematician André Weil apparently compared finding the right definitions in algebraic number theory — which was like carving adamantine rock — to making definitions in the theory of uniform spaces (which he founded), which was like sculpting with snow. In concurrency theory, we are very much at the snow-sculpture end of the spectrum. We lack the kind of external reality, whether it comes from fundamental mathematical objects like the integers, or manifolds, or differential equations, or from physical reality as determined by experiment, which is hard and obdurate, and resistant to our definitions. Is this a necessary feature of our existence, or have we just not yet found the real bedrock?

To design the syntax of a new formal language, we use always the same methodology

Context Free Grammar
Variables
Binders

(In my analogy, always the same programming language)

Is there a better language to define languages?

Motivations

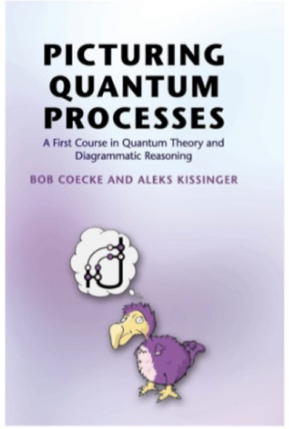
Defining a language like this is not easy at all!
There are many constraints:
The semantics Domain **D** acts as our bedrock

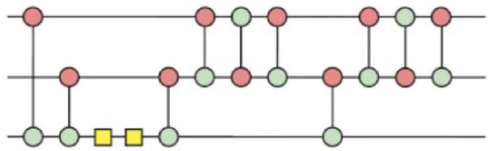
CATEGORY THEORY

Motivations

String diagrams are more and more popular.

Quantum computing



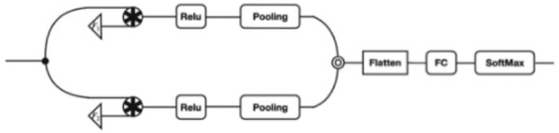


Machine learning

Neural String Diagrams: A Universal Modelling Language for Categorical Deep Learning

Tom Xu and Yoshihiro Maruyama(✉)

School of Computing, Australian National University, Canberra, Australia
{tom.xu,yoshihiro.maruyama}@anu.edu.au

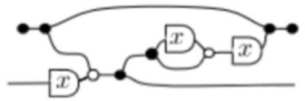


Control theory

A Categorical Semantics of Signal Flow Graphs

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

[1] ENS de Lyon, Université de Lyon, CNRS, INRIA, France
[2] ECS, University of Southampton, UK

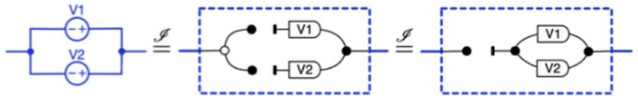


Engineering

String Diagrammatic Electrical Circuit Theory

Guillaume Boisseau
University of Oxford, UK*

Paweł Sobociński
Tallinn University of Technology, Estonia†



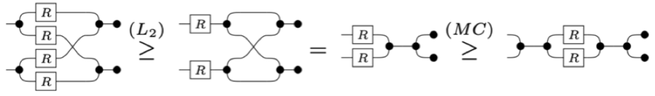
Database

Graphical Conjunctive Queries

Filippo Bonchi
University of Pisa, Italy

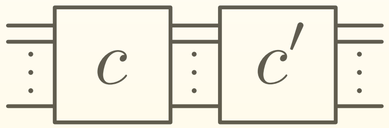
Jens Seeber
IMT School for Advanced Studies Lucca, Italy

Paweł Sobociński
University of Southampton, UK

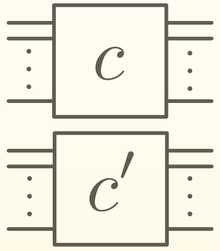


Motivations

String diagrams are circuits that can be composed in



sequence

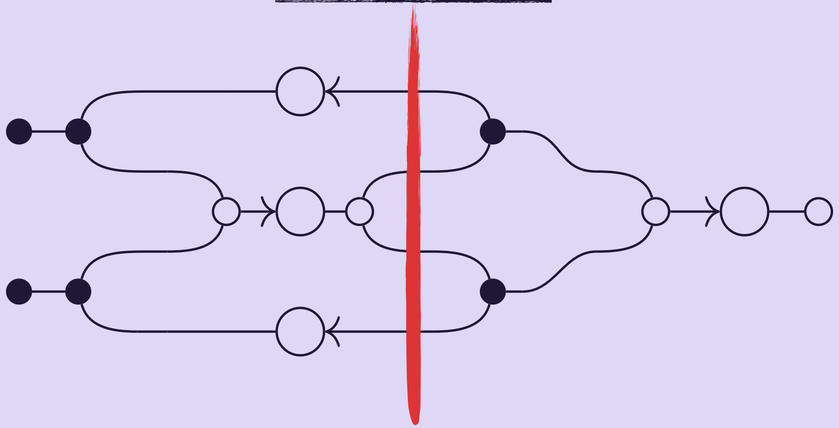


parallel

Intuitive graphical representation easily understandable by engineers (as Hubert suggested...)

But equipped with a formal compositional semantics

POPL 2019

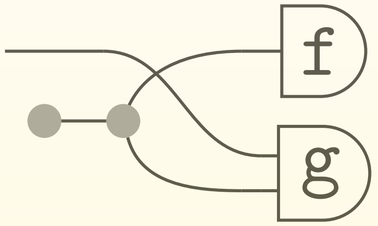


Petri nets are nothing else than linear dynamical systems on the natural numbers
(rather than the reals)

Inf. Comp. 2021

Hoare and Milner synchronisation mechanism satisfy the same algebraic laws

close to Synchronise Hyperedge Replacement



$\nu a_2(\mathtt{f}[a_2/a_1] \,|\, \mathtt{g})$

Motivations

String diagrams correspond to monoidal categories $(\mathbf{C}, \otimes, 1)$.

Motivations

String diagrams correspond to monoidal categories $(\mathbf{C}, \otimes, 1)$.

Sometimes we need *rig categories*: $(\mathbf{C}, \otimes, 1)$ and $(\mathbf{C}, \oplus, 0)$ with $\otimes$ distributing over $\oplus$.

Motivations

String diagrams correspond to monoidal categories $(\mathbf{C}, \otimes, 1)$.

Sometimes we need *rig categories*: $(\mathbf{C}, \otimes, 1)$ and $(\mathbf{C}, \oplus, 0)$ with $\otimes$ distributing over $\oplus$.

Example

The category **Rel** of sets and relations is a rig category, where:

- $R \otimes S$ is the cartesian product
- $R \oplus S$ is the disjoint union

Motivations

String diagrams correspond to monoidal categories $(\mathbf{C}, \otimes, 1)$.

Sometimes we need *rig categories*: $(\mathbf{C}, \otimes, 1)$ and $(\mathbf{C}, \oplus, 0)$ with $\otimes$ distributing over $\oplus$.

Example

The category **Rel** of sets and relations is a rig category, where:

- $R \otimes S$ is the cartesian product
- $R \oplus S$ is the disjoint union

How to depict them?

Motivations

String diagrams correspond to monoidal categories $(\mathbf{C}, \otimes, 1)$.

Sometimes we need *rig categories*: $(\mathbf{C}, \otimes, 1)$ and $(\mathbf{C}, \oplus, 0)$ with $\otimes$ distributing over $\oplus$.

Example

The category **Rel** of sets and relations is a rig category, where:

- $R \otimes S$ is the cartesian product
- $R \oplus S$ is the disjoint union

How to depict them? With *tape diagrams*.

The Calculus of Relations

Syntax

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^{\dagger} \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

The Calculus of Relations

Syntax

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^{\dagger} \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Semantics

Given a *model* $\mathcal{M} = (X, \rho)$, with $\rho(R) \subseteq X \times X$ for all $R \in \Sigma$

The Calculus of Relations

Syntax

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^\dagger \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Semantics

Given a *model* $\mathcal{M} = (X, \rho)$, with $\rho(R) \subseteq X \times X$ for all $R \in \Sigma$

$$\begin{aligned}
[\![R]\!]_{\mathcal{M}} &= \rho(R) & [\![E^\dagger]\!]_{\mathcal{M}} &= \{(x,y) \mid (y,x) \in E\} \\
[\![\top]\!]_{\mathcal{M}} &= X \times X & [\![E_1 \cap E_2]\!]_{\mathcal{M}} &= [\![E_1]\!]_{\mathcal{M}} \cap [\![E_2]\!]_{\mathcal{M}} \\
[\![\bot]\!]_{\mathcal{M}} &= \varnothing & [\![E_1 \cup E_2]\!]_{\mathcal{M}} &= [\![E_1]\!]_{\mathcal{M}} \cup [\![E_2]\!]_{\mathcal{M}} \\
[\![1]\!]_{\mathcal{M}} &= \{(x,x)\} & [\![E_1;E_2]\!]_{\mathcal{M}} &= \{(x,y) \mid \exists z.(x,z) \in [\![E_1]\!]_{\mathcal{M}} \wedge (z,y) \in [\![E_2]\!]_{\mathcal{M}}\}
\end{aligned}$$

The Calculus of Relations

Syntax

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^\dagger \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Semantics

Given a *model* $\mathcal{M} = (X, \rho)$, with $\rho(R) \subseteq X \times X$ for all $R \in \Sigma$

$$\begin{aligned}
[\![R]\!]_{\mathcal{M}} &= \rho(R) & [\![E^\dagger]\!]_{\mathcal{M}} &= \{(x,y) \mid (y,x) \in E\} \\
[\![\top]\!]_{\mathcal{M}} &= X \times X & [\![E_1 \cap E_2]\!]_{\mathcal{M}} &= [\![E_1]\!]_{\mathcal{M}} \cap [\![E_2]\!]_{\mathcal{M}} \\
[\![\bot]\!]_{\mathcal{M}} &= \varnothing & [\![E_1 \cup E_2]\!]_{\mathcal{M}} &= [\![E_1]\!]_{\mathcal{M}} \cup [\![E_2]\!]_{\mathcal{M}} \\
[\![1]\!]_{\mathcal{M}} &= \{(x,x)\} & [\![E_1;E_2]\!]_{\mathcal{M}} &= \{(x,y) \mid \exists z.(x,z) \in [\![E_1]\!]_{\mathcal{M}} \wedge (z,y) \in [\![E_2]\!]_{\mathcal{M}}\}
\end{aligned}$$

Problem (Axiomatic completeness)

$(\forall \mathcal{M} \;.\; [\![E_1]\!]_{\mathcal{M}} = [\![E_2]\!]_{\mathcal{M}}) \Rightarrow E_1 = E_2$?

The Calculus of Relations

Syntax

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^\dagger \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Semantics

Given a *model* $\mathcal{M} = (X, \rho)$, with $\rho(R) \subseteq X \times X$ for all $R \in \Sigma$

$$\begin{array}{rclcrcl}
[\![R]\!]_{\mathcal{M}} & = & \rho(R) & & [\![E^\dagger]\!]_{\mathcal{M}} & = & \{(x,y) \mid (y,x) \in E\} \\
[\![\top]\!]_{\mathcal{M}} & = & X \times X & & [\![E_1 \cap E_2]\!]_{\mathcal{M}} & = & [\![E_1]\!]_{\mathcal{M}} \cap [\![E_2]\!]_{\mathcal{M}} \\
[\![\bot]\!]_{\mathcal{M}} & = & \varnothing & & [\![E_1 \cup E_2]\!]_{\mathcal{M}} & = & [\![E_1]\!]_{\mathcal{M}} \cup [\![E_2]\!]_{\mathcal{M}} \\
[\![1]\!]_{\mathcal{M}} & = & \{(x,x)\} & & [\![E_1;E_2]\!]_{\mathcal{M}} & = & \{(x,y) \mid \exists z.(x,z) \in [\![E_1]\!]_{\mathcal{M}} \wedge (z,y) \in [\![E_2]\!]_{\mathcal{M}}\}
\end{array}$$

Problem (Axiomatic completeness)

$(\forall \mathcal{M} \;.\; [\![E_1]\!]_{\mathcal{M}} = [\![E_2]\!]_{\mathcal{M}}) \Rightarrow E_1 = E_2$?

Answer

No finite axiomatisation [Monk, 1964].

Deconstructing the Regular fragment

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^{\dagger} \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Deconstructing the Regular fragment

$$E \quad ::= \quad R \in \Sigma \;\mid\; 1 \;\mid\; E;E \;\mid\; E^{\dagger} \;\mid\; \top \;\mid\; E \cap E \;\mid\; \bot \;\mid\; E \cup E$$

Deconstructing the Regular fragment

$$E \quad ::= \quad R \in \Sigma \quad | \quad 1 \quad | \quad E;E \quad | \quad E^{\dagger} \quad | \quad \top \quad | \quad E \cap E \quad | \quad \bot \quad | \quad E \cup E$$

Regular logic $(\exists, \wedge)$

Deconstructing the Regular fragment

$$E \quad ::= \quad \boxed{R \in \Sigma \;\mid\; 1 \;\mid\; E;E \;\mid\; E^{\dagger} \;\mid\; \top \;\mid\; E \cap E} \;\mid\; \bot \;\mid\; E \cup E$$

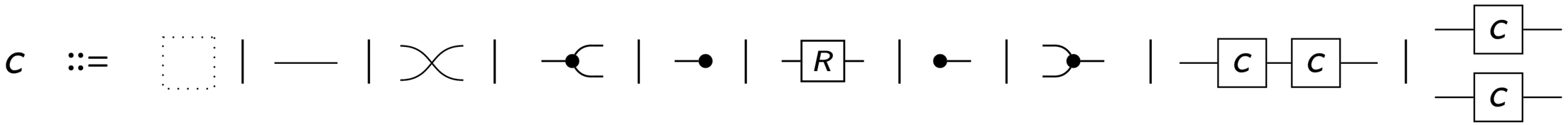
Regular logic $(\exists, \wedge)$

$\Updownarrow$

$(\mathbf{Rel}, \otimes, 1)$

Deconstructing the Regular fragment

Syntax



Deconstructing the Regular fragment

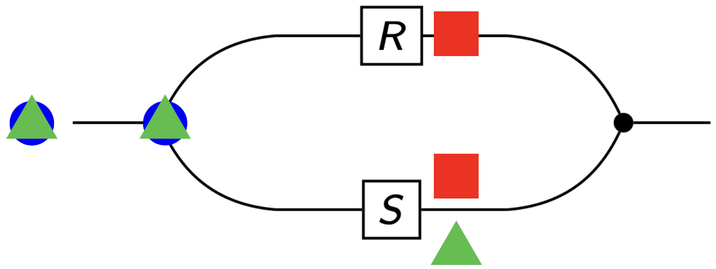
Example

$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}, \qquad R = \{(\bullet, \blacksquare)\}, \quad S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

Deconstructing the Regular fragment

Example

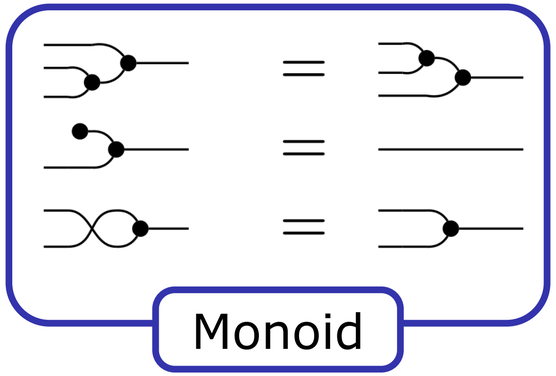
$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}, \qquad R = \{(\bullet, \blacksquare)\}, \quad S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

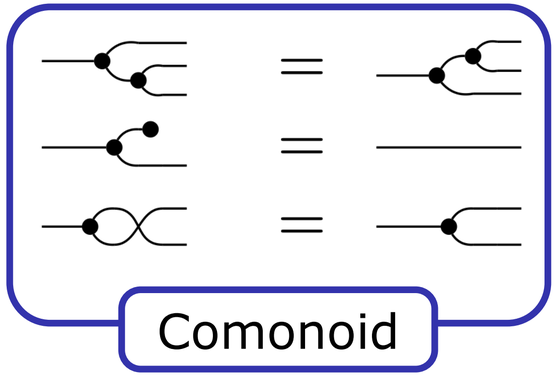


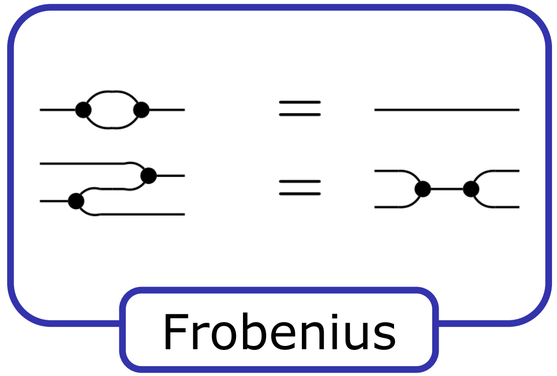
$(\bullet, \blacksquare)$

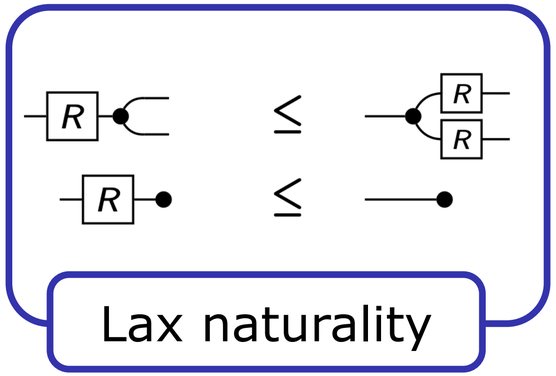
Deconstructing the Regular fragment

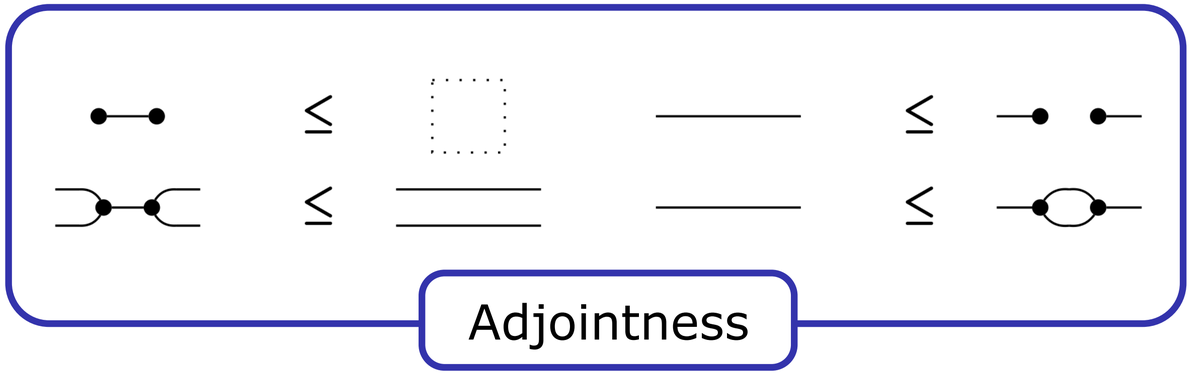
Axioms





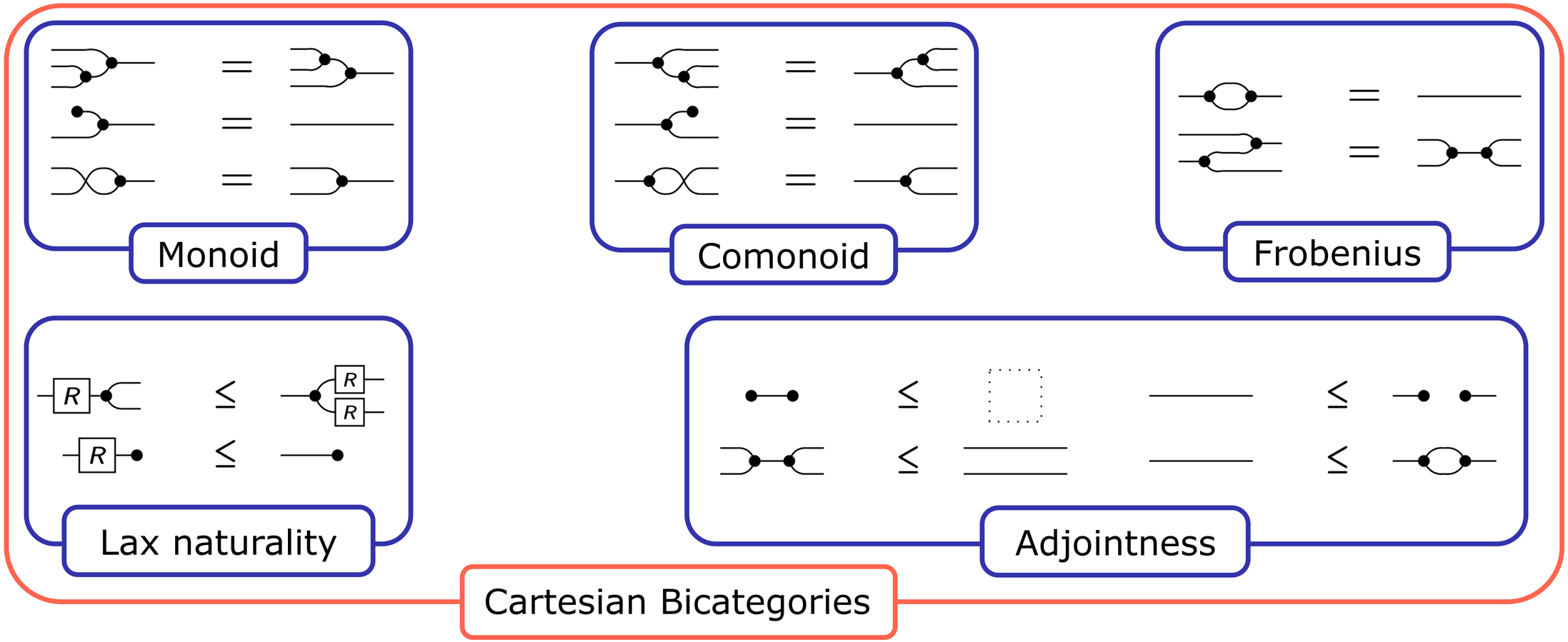






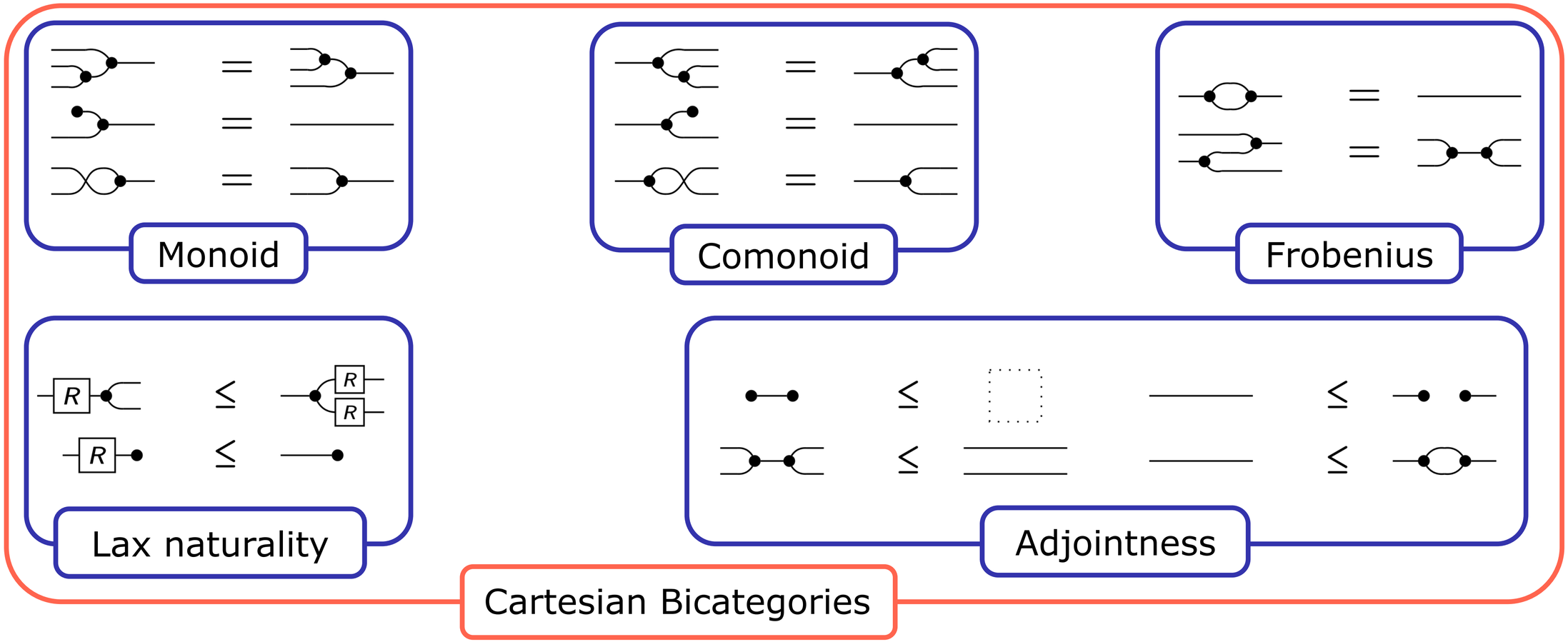
Deconstructing the Regular fragment

Axioms



Deconstructing the Regular fragment

Axioms



Completeness? Yes! [Bonchi, Seeber & Sobocinski, 2018]

Deconstructing the Coherent fragment

$$E \quad ::= \quad R \in \Sigma \;\mid\; 1 \;\mid\; E;E \;\mid\; E^{\dagger} \;\mid\; \top \;\mid\; E \cap E \;\mid\; \bot \;\mid\; E \cup E$$

Regular logic $(\exists, \wedge)$

$\Updownarrow$

$(\mathbf{Rel}, \otimes, 1)$

Deconstructing the Coherent fragment

$$E \quad ::= \quad R \in \Sigma \mid 1 \mid E;E \mid E^\dagger \mid \top \mid E \cap E \mid \bot \mid E \cup E$$

Regular logic $(\exists, \wedge)$

$\Updownarrow$

$(\mathbf{Rel}, \otimes, 1)$

Coherent logic $(\exists, \wedge, \vee)$

Deconstructing the Coherent fragment

$$E \quad ::= \quad R \in \Sigma \mid 1 \mid E;E \mid E^\dagger \mid \top \mid E \cap E \mid \bot \mid E \cup E$$

Regular logic $(\exists, \wedge)$

$\Updownarrow$

$(\mathbf{Rel}, \otimes, 1)$

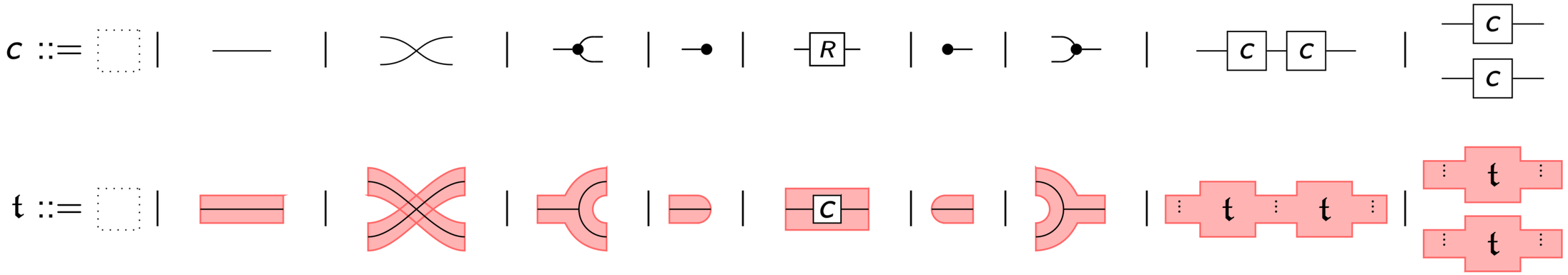
Coherent logic $(\exists, \wedge, \vee)$

$\Updownarrow$

$(\mathbf{Rel}, \otimes, 1, \oplus, 0)$

Tape Diagrams

Syntax



Tape Diagrams

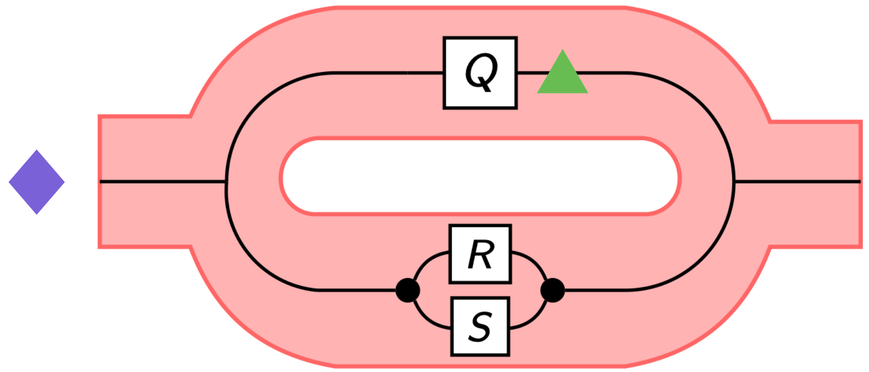
Example

$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}$, $Q = \{(\blacklozenge, \blacktriangle)\}$, $R = \{(\bullet, \blacksquare)\}$, $S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

Tape Diagrams

Example

$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}$, $Q = \{(\blacklozenge, \blacktriangle)\}$, $R = \{(\bullet, \blacksquare)\}$, $S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

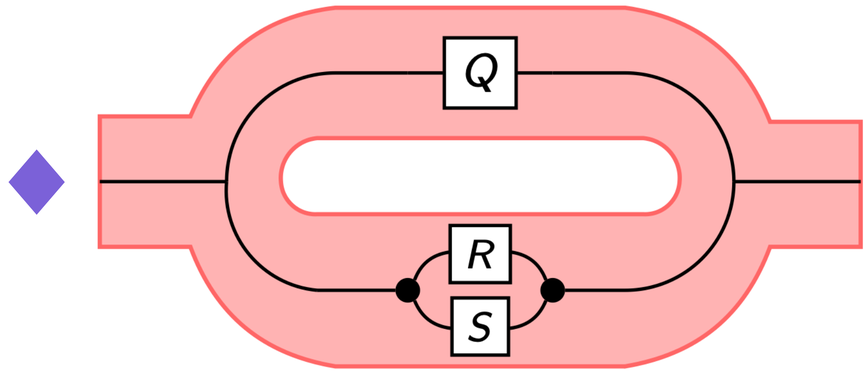


$(\blacklozenge, \blacktriangle)$

Tape Diagrams

Example

$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}, \quad Q = \{(\blacklozenge, \blacktriangle)\}, \quad R = \{(\bullet, \blacksquare)\}, \quad S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

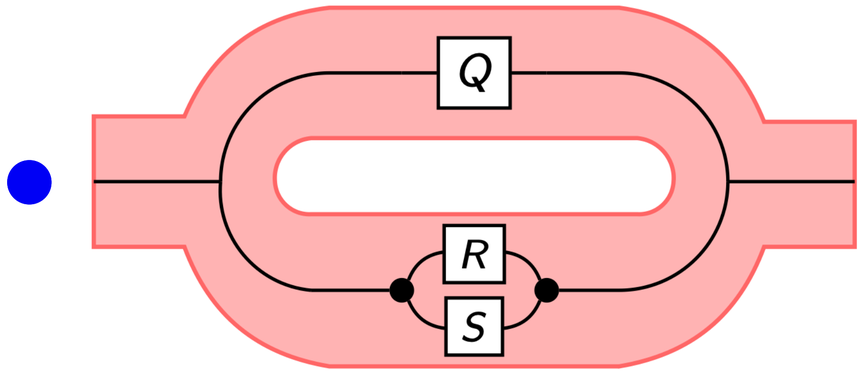


$(\blacklozenge, \blacktriangle)$

Tape Diagrams

Example

$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}$, $Q = \{(\blacklozenge, \blacktriangle)\}$, $R = \{(\bullet, \blacksquare)\}$, $S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$

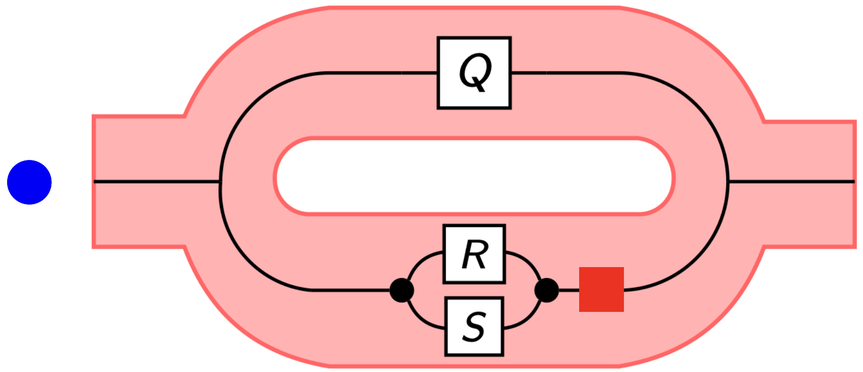


$(\blacklozenge, \blacktriangle)$

Tape Diagrams

Example

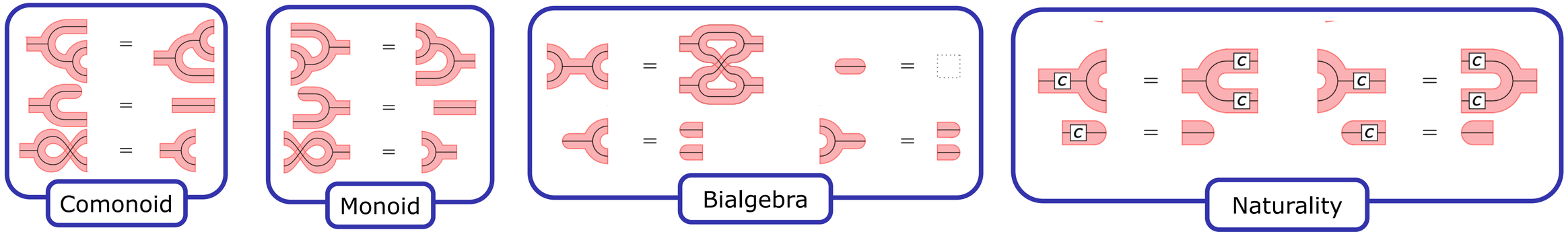
$X = \{\bullet, \blacksquare, \blacktriangle, \blacklozenge\}$, $Q = \{(\blacklozenge, \blacktriangle)\}$, $R = \{(\bullet, \blacksquare)\}$, $S = \{(\bullet, \blacksquare), (\bullet, \blacktriangle)\}$



$(\blacklozenge, \blacktriangle)$

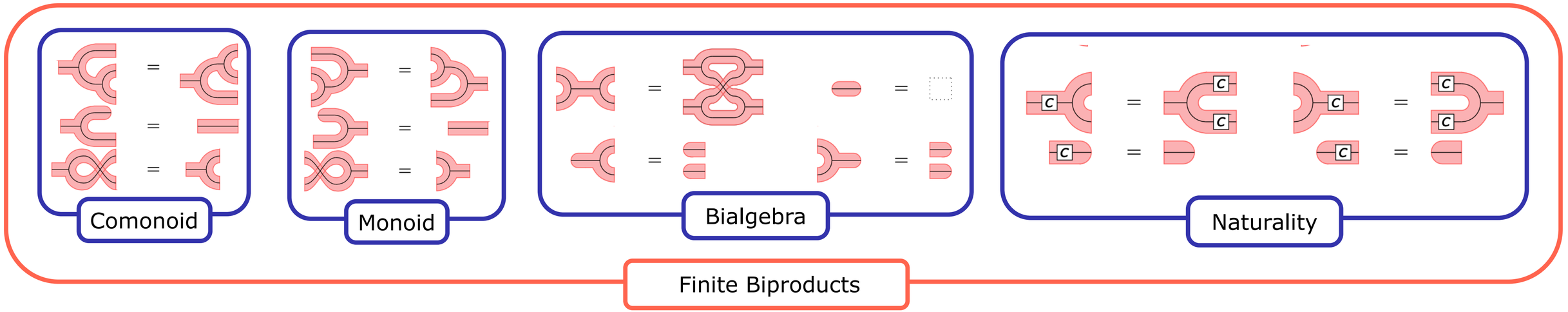
Tape Diagrams

Axioms



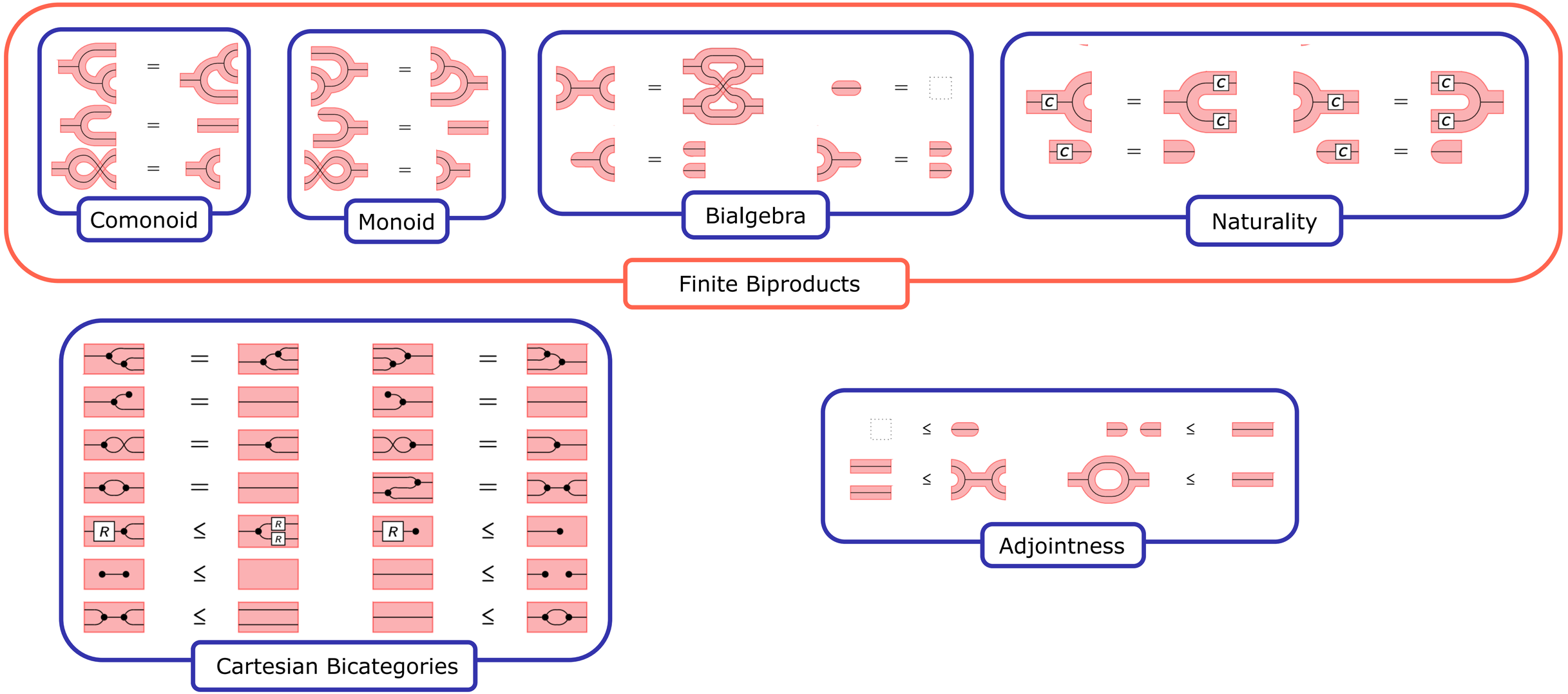
Tape Diagrams

Axioms



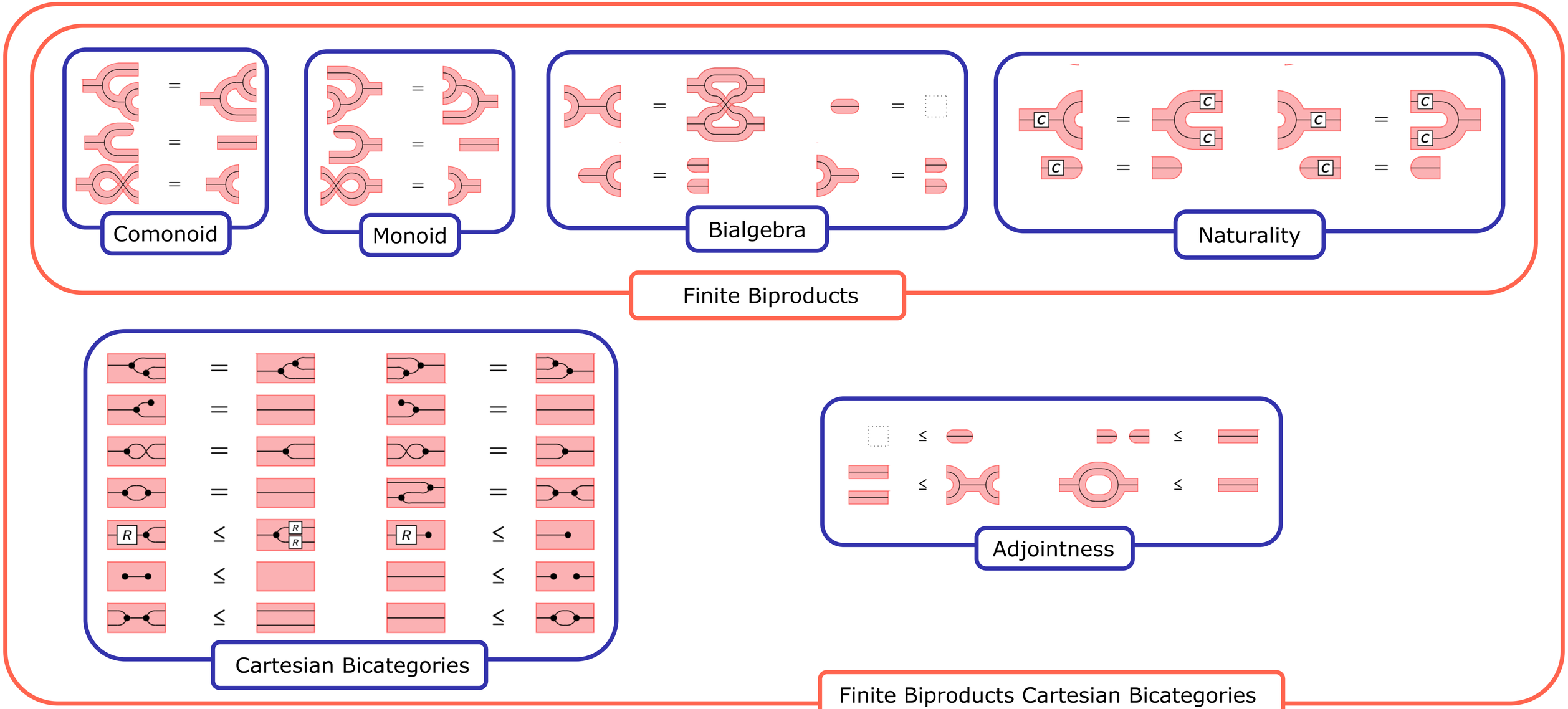
Tape Diagrams

Axioms



Tape Diagrams

Axioms



Tape Diagrams

Completeness?

Tape Diagrams

Completeness?

Yes!

Tape Diagrams

Completeness?

Yes!

Some technical details

- $\mathbf{T}_{\mathbf{CB}_\Sigma}$ is the category of tapes built out of the syntax we saw before
- A model $\mathcal{M}$ is a *rig* functor $\mathbf{T}_{\mathbf{CB}_\Sigma} \to \mathbf{Rel}$

Tape Diagrams

Completeness?

Yes!

Some technical details

- $\mathbf{T}_{\mathbf{CB}_\Sigma}$ is the category of tapes built out of the syntax we saw before
- A model $\mathcal{M}$ is a *rig* functor $\mathbf{T}_{\mathbf{CB}_\Sigma} \to \mathbf{Rel}$

Theorem (Completeness)

For any $\mathfrak{t}, \mathfrak{s}$ *in* $\mathbf{T}_{\mathbf{CB}_\Sigma}$, $(\forall \mathcal{M} \colon \mathbf{T}_{\mathbf{CB}_\Sigma} \to \mathbf{Rel} \;.\; \mathcal{M}(\mathfrak{t}) \subseteq \mathcal{M}(\mathfrak{s})) \implies \mathfrak{t} \leq \mathfrak{s}$.

Not only relations

Quantum computing: ZX-calculus

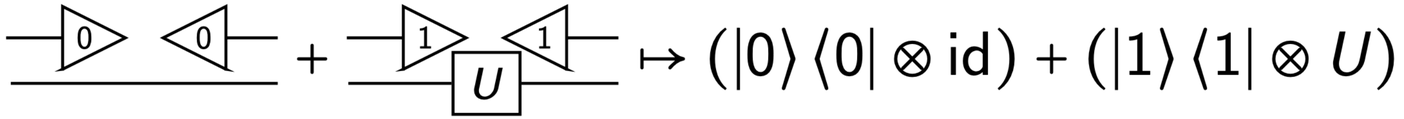
Diagrams denote linear maps (arrows of **FdHilb**)

Not only relations

Quantum computing: ZX-calculus

Diagrams denote linear maps (arrows of **FdHilb**)

Controlled unitary (mixed notation)

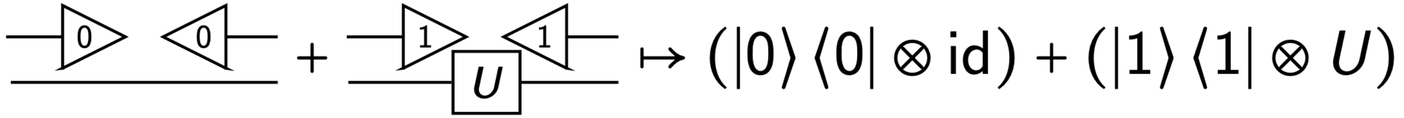


Not only relations

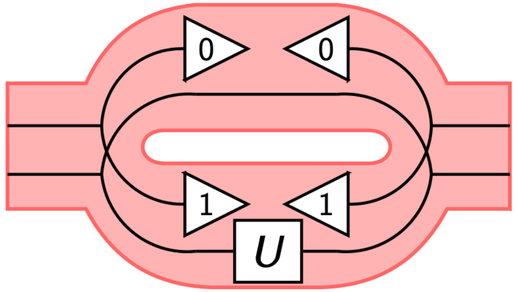
Quantum computing: ZX-calculus

Diagrams denote linear maps (arrows of **FdHilb**)

Controlled unitary (mixed notation)



Controlled unitary (diagrammatic)



Conclusions

Conclusions

Two main contributions

Conclusions

Two main contributions

- Tape diagrams for rig categories

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Future work

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Future work

- ZX-*tapes* for quantum computing

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Future work

- ZX-*tapes* for quantum computing
- Non-deterministic processes as tapes

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Future work

- ZX-*tapes* for quantum computing
- Non-deterministic processes as tapes
- Account for negation in the calculus of relations

Conclusions

Two main contributions

- Tape diagrams for rig categories
- Complete axiomatisation for the (positive) calculus of relations

Future work

- ZX-*tapes* for quantum computing
- Non-deterministic processes as tapes
- Account for negation in the calculus of relations
- Graphical Hoare logic