

Categorical Network Theory

Joe Moeller

`moeller@math.ucr.edu`

UCSB Quantum Algebra and Topology Seminar
2 Oct 2018

Outline

1. Applied Category Theory
2. Categorical Network Theory
3. Network Models
4. Noncommutative Network Models

Applied... Category Theory?

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning
- ▶ Linguistics

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning
- ▶ Linguistics
- ▶ Game Theory

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning
- ▶ Linguistics
- ▶ Game Theory
- ▶ Dynamical Systems

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning
- ▶ Linguistics
- ▶ Game Theory
- ▶ Dynamical Systems
- ▶ Network Theory, e.g. electrical circuits, control theory

Applied... Category Theory?

- ▶ *Isn't that an oxymoron?*
- ▶ *Applied to other math?*
- ▶ *Logic*
- ▶ *Computer science, e.g. functional programming.*
- ▶ Quantum mechanics/computation
- ▶ Machine Learning
- ▶ Linguistics
- ▶ Game Theory
- ▶ Dynamical Systems
- ▶ Network Theory, e.g. electrical circuits, control theory

Who does this stuff?

Who does this stuff?

Some people:

- ▶ John Baez
- ▶ Bob Coecke
- ▶ Brendan Fong
- ▶ Jules Hedges
- ▶ Martha Lewis
- ▶ Mehrnoosh Sadrzadeh
- ▶ Pawel Sobocinski
- ▶ David Spivak
- ▶ Jamie Vicary

Who does this stuff?

Some people:

- ▶ John Baez
- ▶ Bob Coecke
- ▶ Brendan Fong
- ▶ Jules Hedges
- ▶ Martha Lewis
- ▶ Mehrnoosh Sadrzadeh
- ▶ Pawel Sobocinski
- ▶ David Spivak
- ▶ Jamie Vicary

Blockchain companies:

- ▶ Pyroflex/RChain
- ▶ Statebox

Two Themes

Compositionality: when an attribute of something can be determined from that attribute of its parts and the way they interact.

Two Themes

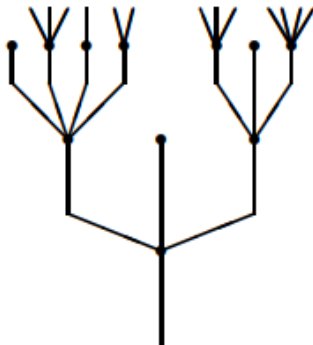
Compositionality: when an attribute of something can be determined from that attribute of its parts and the way they interact.

Functorial semantics: Syntax \rightarrow Semantics

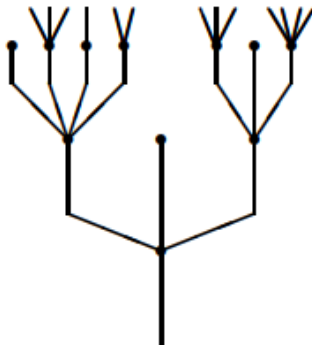
Categorical Perspectives on Network Theory

- ▶ Wiring Diagrams
- ▶ Decorated Cospans
- ▶ Petri Nets
- ▶ Network Models

Operads

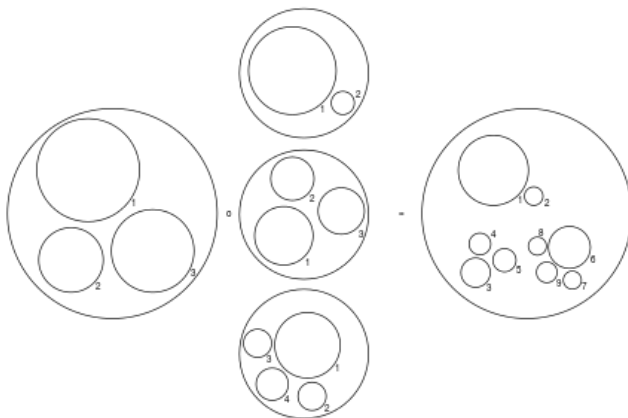


Operads



Example: real functions

Little Discs Operad

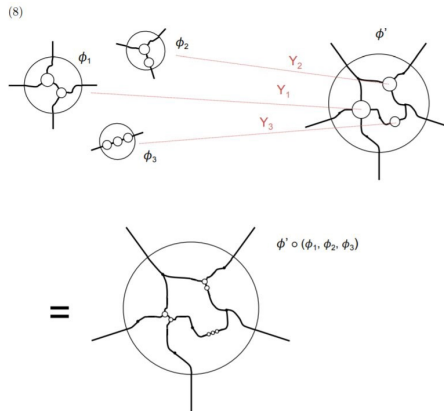


Underlying Operad

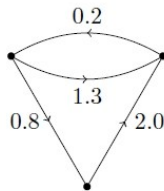
Given a symmetric monoidal category (\mathbf{C}, \otimes) , one can get an operad $\text{op}(\mathbf{C})$, called the **underlying operad**, with

- ▶ objects of \mathbf{C} as types
- ▶ $\text{op}(\mathbf{C})(c_1, \dots, c_n; c) = \text{Hom}_{\mathbf{C}}(c_1 \otimes \dots \otimes c_n, c)$

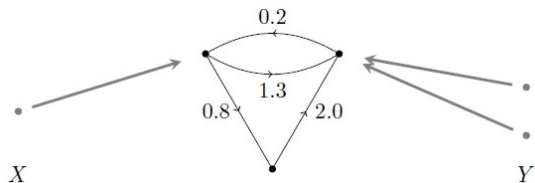
Wiring Diagrams



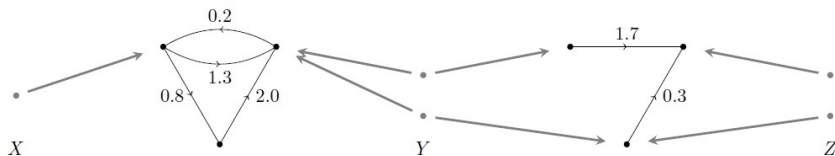
Decorated Cospans



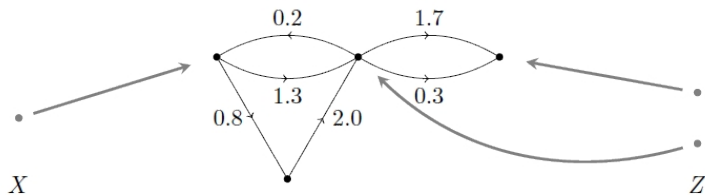
Decorated Cospans



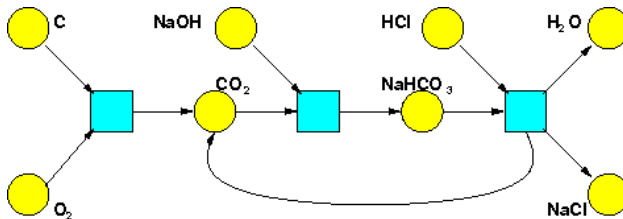
Decorated Cospans



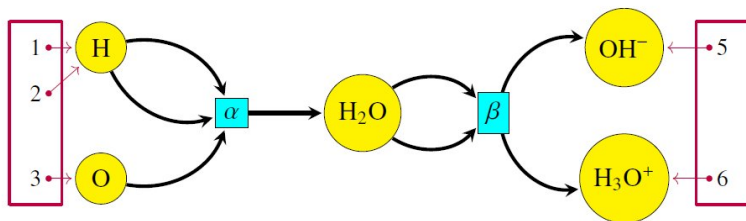
Decorated Cospans



Petri Nets



Open Petri Nets



CASCADE

Complex Adaptive System Composition And Design Environment

- ▶ The goal of CASCADE is to provide a unified view of system behavior, allowing understanding of these complex interactions and a formal language for complex adaptive system composition and design. This unified view of system behavior, enabled by appropriate mathematical foundations, may also enable adaptation to unanticipated environments using arbitrary system components by providing a framework to dynamically identify and correct deficient system capabilities.

Constructing Network Operads

- ▶ Start with your favorite lax symmetric monoidal functor $F: S \rightarrow \text{Cat}$
- ▶ apply the symmetric monoidal Grothendieck construction to get the symmetric monoidal category $\int F$ with \otimes_F
- ▶ Let O_F be the endomorphism operad $\text{op}(\int F)$

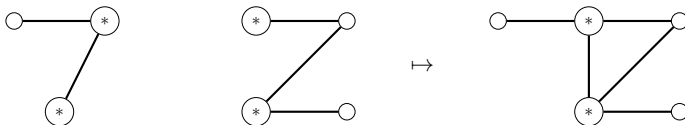
Theorem (M)

The composite

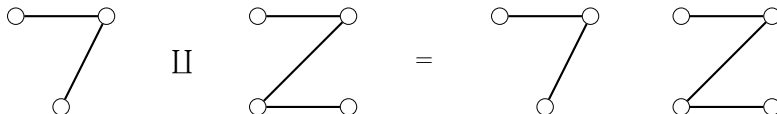
$$\text{NetMod} \xrightarrow{\int} \text{SSMC} \xrightarrow{\text{op}(-)} \text{Op}$$

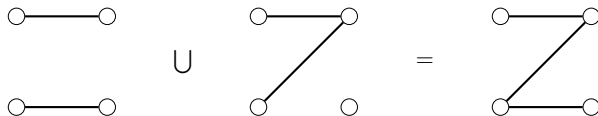
constructs a network operad for each network model.

Graphs can be combined to create bigger graphs by identifying some of the vertices

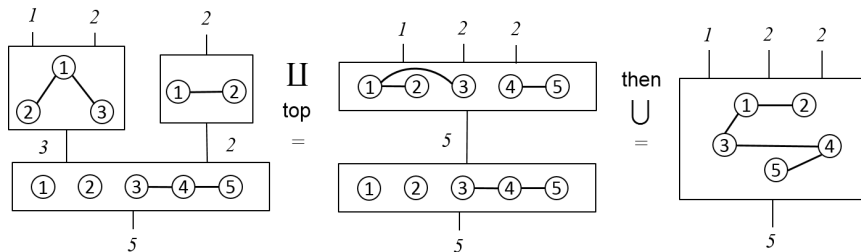


We choose to examine these as combinations of a few simpler operations

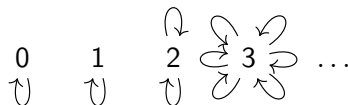




We want to construct an operad that captures these operations



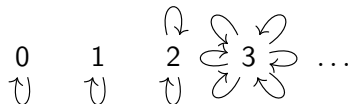
The Permutation Groupoid



Definition

Let S denote the category with finite sets $\mathbf{n} = \{1, \dots, n\}$ as objects, and bijections for morphisms.

The Permutation Groupoid



Definition

Let S denote the category with finite sets $\mathbf{n} = \{1, \dots, n\}$ as objects, and bijections for morphisms.

Another way to see it is $S = \coprod_{n \in \mathbb{N}} S_n$.

The Permutation Groupoid

S is a symmetric monoidal category with $+$ where

- ▶ $n + m$ is exactly what you think
- ▶ $\sigma + \tau$ looks like

$$\begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} + \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} = \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array} \begin{array}{c} \diagup \diagdown \\ \diagdown \diagup \end{array}$$

Network Models

Definition

A **Network Model** is a lax symmetric monoidal functor $F: S \rightarrow \text{Cat}$. Let NetMod denote the category of network models.

Network Models

Definition

A **Network Model** is a lax symmetric monoidal functor $F: S \rightarrow \text{Cat}$. Let NetMod denote the category of network models.

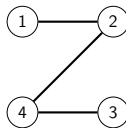
Why this definition?

Simple Graphs

A simple graph with vertex set $\mathbf{n} = \{1, \dots, n\}$ is a collection of subsets of \mathbf{n} , each of which have 2 elements.

Simple Graphs

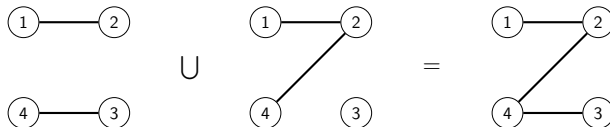
A simple graph with vertex set $\mathbf{n} = \{1, \dots, n\}$ is a collection of subsets of \mathbf{n} , each of which have 2 elements. For example, this graph on $\mathbf{4}$



is the set $\{\{1, 2\}, \{2, 4\}, \{3, 4\}\}$ in this setting.

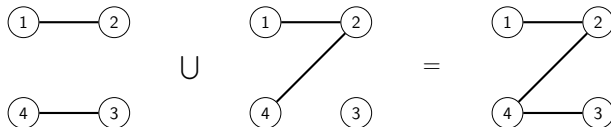
Graph Monoids

Defining graphs this way allows us to take unions of graphs



Graph Monoids

Defining graphs this way allows us to take unions of graphs



so the set of simple graphs on \mathbf{n} , denoted $\text{SG}(\mathbf{n})$, is a monoid with operation \cup .

S_n acts on the monoid $SG(\mathbf{n})$ by permuting the vertices.

S_n acts on the monoid $\text{SG}(\mathbf{n})$ by permuting the vertices. So we have a functor for each $n \in \mathbb{N}$

$$\text{SG}_n: S_n \rightarrow \text{Mon}$$

S_n acts on the monoid $\text{SG}(\mathbf{n})$ by permuting the vertices. So we have a functor for each $n \in \mathbb{N}$

$$\text{SG}_n: S_n \rightarrow \text{Mon}$$

Putting them all together gives a functor

$$\text{SG}: S \rightarrow \text{Mon}$$

Given two graphs, g in $SG(\mathbf{n})$ and h in $SG(\mathbf{m})$, the disjoint union $g \sqcup h$ is a graph in $SG(\mathbf{n} + \mathbf{m})$.

Given two graphs, g in $\mathbf{SG}(\mathbf{n})$ and h in $\mathbf{SG}(\mathbf{m})$, the disjoint union $g \sqcup h$ is a graph in $\mathbf{SG}(\mathbf{n} + \mathbf{m})$. This gives a map

$$\sqcup: \mathbf{SG}(\mathbf{n}) \times \mathbf{SG}(\mathbf{m}) \rightarrow \mathbf{SG}(\mathbf{n} + \mathbf{m})$$

Given two graphs, g in $\mathbf{SG}(\mathbf{n})$ and h in $\mathbf{SG}(\mathbf{m})$, the disjoint union $g \sqcup h$ is a graph in $\mathbf{SG}(\mathbf{n} + \mathbf{m})$. This gives a map

$$\sqcup: \mathbf{SG}(\mathbf{n}) \times \mathbf{SG}(\mathbf{m}) \rightarrow \mathbf{SG}(\mathbf{n} + \mathbf{m})$$

which makes \mathbf{SG} a symmetric lax monoidal functor

$$(\mathbf{SG}, \sqcup): (\mathbf{S}, +) \rightarrow (\mathbf{Cat}, \times)$$

Combinatorics Captured by the Definition

1. $e_n \cup g = g = g \cup e_n$
2. $g_1 \cup (g_2 \cup g_3)$
3. $\sigma(g_1 \cup g_2) = \sigma g_1 \cup \sigma g_2$
4. $\sigma e_n = e_n$
5. $(\sigma_1 \sigma_2)g = \sigma_1(\sigma_2 g)$
6. $1(g) = g$

Combinatorics Captured by the Definition

1. $e_n \cup g = g = g \cup e_n$
2. $g_1 \cup (g_2 \cup g_3)$
3. $\sigma(g_1 \cup g_2) = \sigma g_1 \cup \sigma g_2$
4. $\sigma e_n = e_n$
5. $(\sigma_1 \sigma_2)g = \sigma_1(\sigma_2 g)$
6. $1(g) = g$
7. $(g_1 \cup g_2) \sqcup (h_1 \cup h_2) = (g_1 \sqcup h_1) \cup (g_2 \sqcup h_2)$
8. $e_m \sqcup e_n = e_{m+n}$
9. $\sigma g \sqcup \tau h = (\sigma + \tau)(g \sqcup h)$
10. $g_1 \sqcup (g_2 \sqcup g_3) = (g_1 \sqcup g_2) \sqcup g_3$
11. $e_0 \sqcup g = g \sqcup e_0$
12. $B_{m,n}(h \sqcup g) = g \sqcup h$

The Grothendieck Construction

The Grothendieck construction takes a pseudofunctor

$$F: \mathbf{C}^{op} \rightarrow \mathbf{Cat}$$

The Grothendieck Construction

The Grothendieck construction takes a pseudofunctor

$$F: \mathbf{C}^{op} \rightarrow \mathbf{Cat}$$

and produces a category fibred over \mathbf{C}

$$\begin{array}{c} \int F \\ \downarrow \\ \mathbf{C} \end{array}$$

The Grothendieck Construction

Given a functor $F: \mathbf{C} \rightarrow \mathbf{Cat}$ the Grothendieck construction gives a category $\int F$ where

- ▶ objects are pairs (c, x) where
 - ▶ c is an object in \mathbf{C}
 - ▶ x is an object in Fc
- ▶ morphisms are $(f, g): (c, x) \rightarrow (d, y)$ where
 - ▶ $f: c \rightarrow d$ in \mathbf{C}
 - ▶ $g: Ff(x) \rightarrow y$ in Fd
- ▶ composition is given by

$$(f, g) \circ (f', g') = (f \circ f', g \circ Ff(g'))$$

The Monoidal Grothendieck Construction

Let (\mathbf{C}, \otimes) be a monoidal category, and $F: \mathbf{C} \rightarrow \mathbf{Cat}$ a lax monoidal functor, with $\Phi_{c,d}: Fc \times Fd \rightarrow F(c \otimes d)$.

The Monoidal Grothendieck Construction

Let (\mathbf{C}, \otimes) be a monoidal category, and $F: \mathbf{C} \rightarrow \mathbf{Cat}$ a lax monoidal functor, with $\Phi_{c,d}: Fc \times Fd \rightarrow F(c \otimes d)$.

Then we can define a monoidal structure on $\int F$ by

$$(c, x) \otimes_F (d, y) = (c \otimes d, \Phi_{c,d}(x, y))$$

and

$$(f, g) \otimes_F (f', g') = (f \otimes f', \Phi_{d,d'}(g, g'))$$

(Braided/Symmetric) Monoidal Grothendieck Construction

Theorem (M, Vasilakopoulou)

If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a lax monoidal functor, there is a natural way to define a monoidal structure on $\int F$.

(Braided/Symmetric) Monoidal Grothendieck Construction

Theorem (M, Vasilakopoulou)

If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a lax monoidal functor, there is a natural way to define a monoidal structure on $\int F$.

Theorem (M, Vasilakopoulou)

If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a braided lax monoidal functor, there is a natural way to define a braided monoidal structure on $\int F$.

(Braided/Symmetric) Monoidal Grothendieck Construction

Theorem (M, Vasilakopoulou)

If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a lax monoidal functor, there is a natural way to define a monoidal structure on $\int F$.

Theorem (M, Vasilakopoulou)

If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a braided lax monoidal functor, there is a natural way to define a braided monoidal structure on $\int F$.

Theorem (M, Vasilakopoulou)

*If $F: \mathbf{C} \rightarrow \mathbf{Cat}$ is a symmetric lax monoidal functor, there is a natural way to define a symmetric monoidal structure on $\int F$. We call this the **symmetric monoidal Grothendieck construction**.*

Constructing Network Operads

- ▶ Start with your favorite network model, i.e. lax symmetric monoidal functor, $F: S \rightarrow \text{Cat}$, for example SG

Constructing Network Operads

- ▶ Start with your favorite network model, i.e. lax symmetric monoidal functor, $F: S \rightarrow \text{Cat}$, for example SG
- ▶ apply the symmetric monoidal Grothendieck construction to get the symmetric monoidal category $\int F$ with \otimes_F

Constructing Network Operads

- ▶ Start with your favorite network model, i.e. lax symmetric monoidal functor, $F: S \rightarrow \text{Cat}$, for example SG
- ▶ apply the symmetric monoidal Grothendieck construction to get the symmetric monoidal category $\int F$ with \otimes_F
- ▶ Let O_F be the endomorphism operad $\text{op}(\int F)$

Examples of Network Models

- ▶ Multigraphs
- ▶ Directed Graphs
- ▶ Partitions
- ▶ Graphs with colored vertices
- ▶ Petri Nets

Examples of Network Models

- ▶ Multigraphs
- ▶ Directed Graphs
- ▶ Partitions
- ▶ Graphs with colored vertices
- ▶ Petri Nets
- ▶ Graphs with edges weighted by a monoid

Range-Limited Networks

Let $L_k(\mathbf{n})$ denote the set of k -limited networks with vertex set \mathbf{n} .

Range-Limited Networks

Let $L_k(\mathbf{n})$ denote the set of k -limited networks with vertex set \mathbf{n} .
This forms an algebra of the simple graphs network model.

Networks of Bounded Degree

The **degree of a vertex** in an undirected graph is the number of edges containing that vertex.

Networks of Bounded Degree

The **degree of a vertex** in an undirected graph is the number of edges containing that vertex. The **degree of a graph** is the highest degree of any of its vertices.

Networks of Bounded Degree

The **degree of a vertex** in an undirected graph is the number of edges containing that vertex. The **degree of a graph** is the highest degree of any of its vertices.

Let $B_k(\mathbf{n})$ be the set of graphs with degree $\leq k$.

Eckmann-Hilton for Network Models

Let (F, \sqcup) be a network model, $a \in F(n)$ and $b \in F(m)$. Then

$$\begin{aligned}(a \sqcup e_m) \cup (e_n \sqcup b) &= (a \cup e_n) \sqcup (e_m \cup b) \\ &= (e_n \cup a) \sqcup (b \cup e_m) \\ &= (e_n \sqcup b) \cup (a \sqcup e_m)\end{aligned}$$

Eckmann-Hilton for Network Models

Let (F, \sqcup) be a network model, $a \in F(n)$ and $b \in F(m)$. Then

$$\begin{aligned}(a \sqcup e_m) \cup (e_n \sqcup b) &= (a \cup e_n) \sqcup (e_m \cup b) \\ &= (e_n \cup a) \sqcup (b \cup e_m) \\ &= (e_n \sqcup b) \cup (a \sqcup e_m)\end{aligned}$$

So there are relative commutativity relations that must hold in each monoid of a network model.

Free Network Models

Theorem (M)

The functor $\text{NetMod} \rightarrow \text{Mon}$ defined by $F \mapsto F(2)$ has a left adjoint $\Gamma_{-, \text{Mon}} : \text{Mon} \rightarrow \text{NetMod}$, with $\Gamma_{M, \text{Mon}}(\mathbf{n}) = KG_{n,2}(M)$.

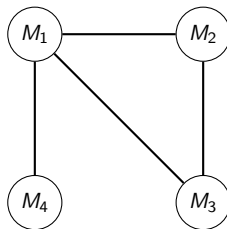
Free Network Models

Theorem (M)

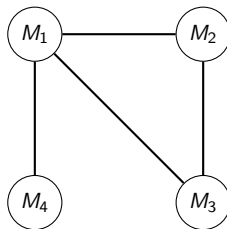
The functor $\text{NetMod} \rightarrow \text{Mon}$ defined by $F \mapsto F(2)$ has a left adjoint $\Gamma_{-, \text{Mon}}: \text{Mon} \rightarrow \text{NetMod}$, with $\Gamma_{M, \text{Mon}}(\mathbf{n}) = KG_{n,2}(M)$.

This network model is able to remember the order in which edges were added to a network

Graph Products of Monoids



Graph Products of Monoids



$$G(\{M_i\}) = \coprod M_i / R$$

Graph Products

Theorem (M)

Graph products are isomorphic to the colimit of the diagram as described.

Kneser Graphs

The Kneser graph $KG_{n,m}$ has vertex set $\binom{n}{m}$, the set of m -element subsets of an n -element set, and an edge between two vertices if they are disjoint subsets.

Kneser Graphs

The Kneser graph $KG_{n,m}$ has vertex set $\binom{n}{m}$, the set of m -element subsets of an n -element set, and an edge between two vertices if they are disjoint subsets.

The graphs $KG_{n,2}$ are perfect for describing the disjointness-commutativity that network models demand!




Free Network Models

Theorem (M)

Given a monoid M , there is a network model $\Gamma_{M,\text{Mon}}$ with monoids given by Kneser graph products of M ,

$$\Gamma_{M,\text{Mon}}(\mathbf{n}) = KG_{n,2}(M).$$

This gives the left adjoint to the functor $F \mapsto F(2)$.

-  J. C. Baez, J. D. Foley, J. Moeller and B. S. Pollard, Network Models, Preprint, Available as arXiv:1711.00037.
-  J. Moeller, Noncommutative Network Models, Preprint, Available as arXiv:1804.07402.
-  J. Moeller and C. Vasilakopoulou, Monoidal Grothendieck Construction, Preprint, Available as arXiv:1809.00727.