

# Copy-Cat Strategies and Information Flow in Physics, Geometry, Logic and Computation

Samson Abramsky  
Oxford University Computing Laboratory

## Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?
- Some themes

## Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# Introduction

# How to Beat a Grand-Master

## Introduction

- **How to Beat a Grand-Master**
- Does Copy-Cat still work here?
- Some themes

## Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# How to Beat a Grand-Master

## Introduction

- **How to Beat a Grand-Master**
- Does Copy-Cat still work here?
- Some themes

## Logic

From Proof Nets to Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

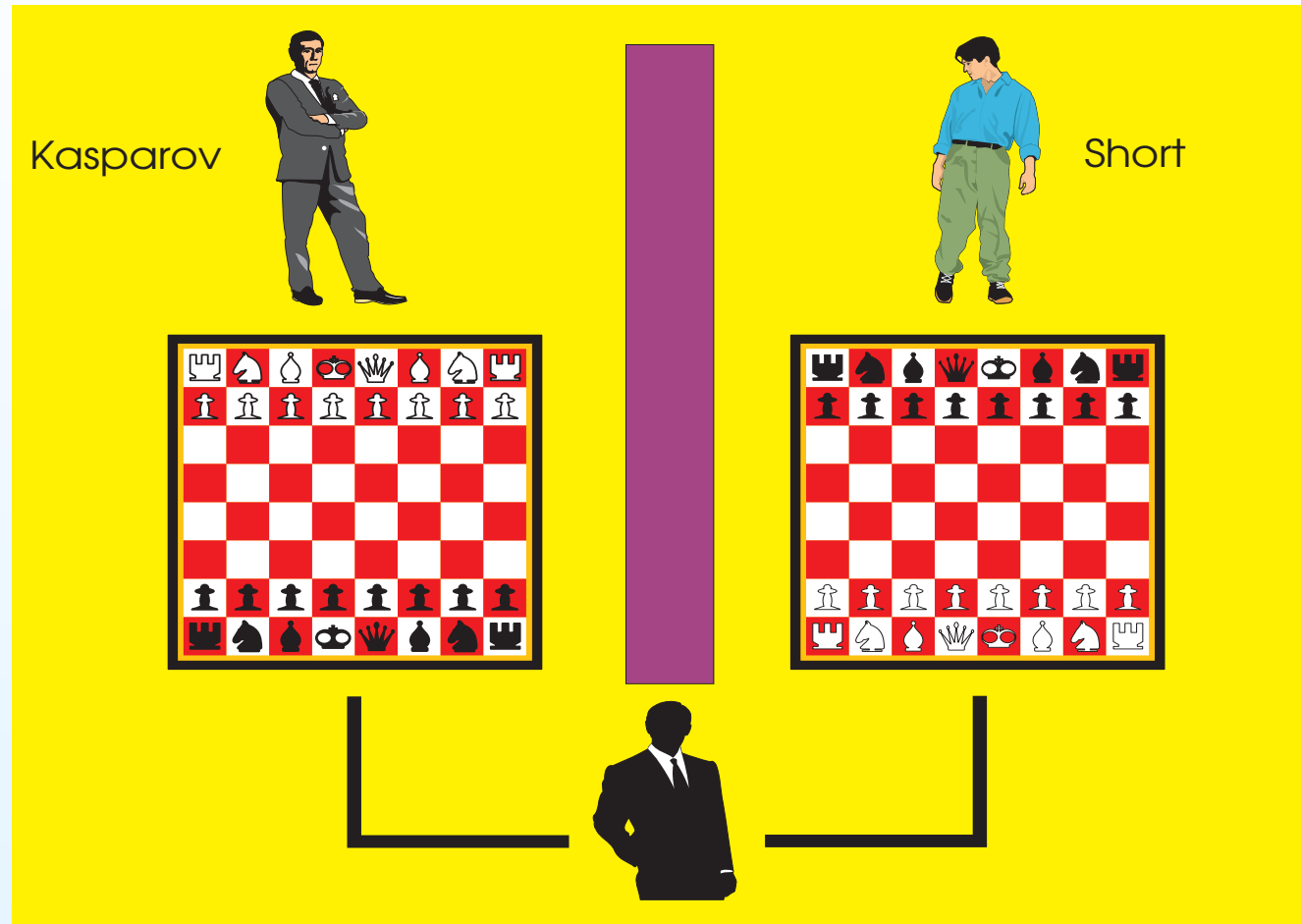
Composition

Functional Computation: The Planar  $\lambda$ -calculus

Functional Computation: Non-Planar Combinators

Logic of Quantum Information Flow

Cloning



## The Copy-Cat Strategy

# Does Copy-Cat still work here?

## Introduction

- How to Beat a Grand-Master
- **Does Copy-Cat still work here?**
- Some themes

## Logic

From Proof Nets to Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional Computation: The Planar  $\lambda$ -calculus

Functional Computation: Non-Planar Combinators

Logic of Quantum Information Flow

Cloning

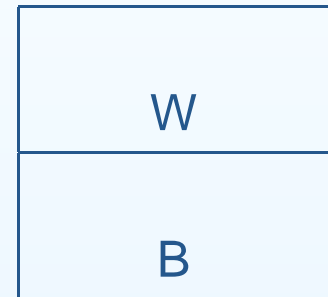
Kasparov



Short



Short



# Some themes

## Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?
- **Some themes**

## Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?
- **Some themes**

### Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?

- **Some themes**

### Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric
- Correspondence between **interactive** and **geometric** views of the same phenomena

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?

- **Some themes**

### Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric
- Correspondence between **interactive** and **geometric** views of the same phenomena
- Emergence

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?

- **Some themes**

### Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric
- Correspondence between **interactive** and **geometric** views of the same phenomena
- Emergence
- Copy-cat vs. Cloning:

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?

- **Some themes**

### Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

### Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric
- Correspondence between **interactive** and **geometric** views of the same phenomena
- Emergence
- Copy-cat vs. Cloning:
  - Linear vs. Classical Logic
  - Quantum vs. Classical Physics
  - Linear-time vs. Computationally universal

## Some themes

### Introduction

- How to Beat a Grand-Master
- Does Copy-Cat still work here?

- **Some themes**

### Logic

#### From Proof Nets to Diagram Algebras

#### Temperley-Lieb Algebra

#### Characterizing Planarity

### Composition

#### Functional Computation: The Planar $\lambda$ -calculus

#### Functional Computation: Non-Planar Combinators

#### Logic of Quantum Information Flow

### Cloning

- Common fundamental structures of interaction: logical, computational, physical, geometric
- Correspondence between **interactive** and **geometric** views of the same phenomena
- Emergence
- Copy-cat vs. Cloning:
  - Linear vs. Classical Logic
  - Quantum vs. Classical Physics
  - Linear-time vs. Computationally universal
- Many pictures — serious mathematical structure underneath! (Joyal, Street, Kelly, Penrose, ...)
- Game aspect not emphasized — more at primitive interaction level ('Go!').

## Introduction

## Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- Understanding Proof Nets Interactively
- Semantics of MLL Proofs
- Semantics: Soundness and Completeness
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

From Proof Nets to Diagram Algebras

## Temperley-Lieb Algebra

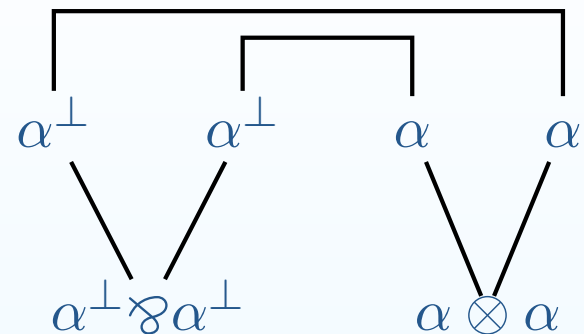
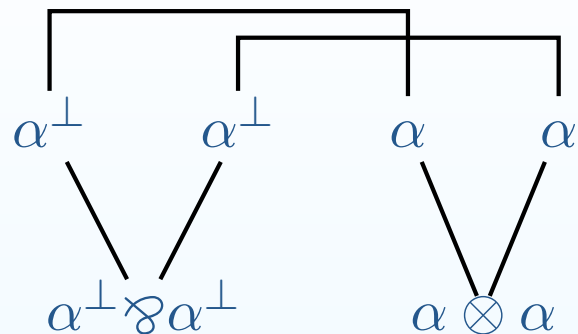
## Characterizing Planarity

## Composition

## Functional

# Logic

## Multiplicative Proof Structures



The essential information in a (cut-free) proof in MLL is the axiom links.

Accordingly, we define a **proof structure** on a sequent  $\Gamma$  to be a fixpoint-free involution  $f$  (so  $f^2 = 1$  and  $f(a) \neq a$ ) on its occurrences of literals such that if  $f(a) = b$ ,  $l(a) = l(b)^\perp$ .

# From Proof Nets to Semantics

## Introduction

## Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- Understanding Proof Nets Interactively
- Semantics of MLL Proofs
- Semantics: Soundness and Completeness
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

## From Proof Nets to Diagram Algebras

## Temperley-Lieb Algebra

## Characterizing Planarity

## Composition

## Functional

Note that proof structures as we have defined them are simply certain permutations acting on finite sets (of literals). This leads to the following compositional interpretation of formulas by finite sets, and of proofs by permutations on these sets.

- A literal is interpreted by a one-point set; Tensor and Par by disjoint union. A sequent is treated like the Par of its formulas. Thus the set  $|\Gamma|$  associated to a sequent is in bijection with its set of occurrences of literals.

# Assignment of Permutations to Sequent Proofs

## Axiom

$$\frac{}{\vdash a, a^\perp} \text{Id}$$

## Multiplicatives

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B} \otimes \quad \frac{\Gamma, A, B}{\vdash \Gamma, A \wp B} \wp$$

- Axiom: assign the transposition  $a \leftrightarrow a^\perp$
- Tensor: assign the disjoint union of the two permutations
- Par: assign the same permutation!

## MLL Proof Nets

Which proof structures really come from proofs in MLL?

**Switching Graphs:** A **switching**  $S$  of  $\Gamma$  assigns  $L$  or  $R$  to each occurrence of  $\wp$ . Given a sequent  $\Gamma$ , a proof structure  $f$ , and a switching  $S$ , the **switching graph**  $G(\Gamma, f, S)$  has:

- subformula occurrences in  $\Gamma$  as vertices;
- an edge connecting  $A$  to  $A \otimes B$  and an edge connecting  $B$  to  $A \otimes B$  for each occurrence of  $A \otimes B$ ;
- an edge connecting  $A$  to  $A \wp B$  if  $S$  assigns  $L$  to  $A \wp B$ , and an edge connecting  $B$  to  $A \wp B$  if  $S$  assigns  $R$  to  $A \wp B$ ;
- an edge connecting literal occurrences  $a$  and  $b$  if  $f(a) = b$ .

**The Danos-Regnier criterion:** A proof-structure  $f$  for  $\Gamma$  is an MLL proof-net if for every switching  $S$ ,  $G(\Gamma, f, S)$  is **acyclic and connected**.

# Results on Proof Nets

Introduction

Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- **Results on Proof Nets**
- Understanding Proof Nets Interactively
- Semantics of MLL Proofs
- Semantics: Soundness and Completeness
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

From Proof Nets to Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional

Computation: The Copy-Cat Strategies and Information Flow  
Planar  $\lambda$ -calculus

Every sequent proof in MLL canonically maps to a proof structure.

**Proposition 1 (Soundness)** *The proof structures arising from sequent proofs are proof nets.*

**Theorem 2 (Sequentialization Theorem)** *Every proof net arises from a sequent proof.*

This is the **Geometric Criterion**.

# Understanding Proof Nets Interactively

## Introduction

## Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- **Understanding Proof Nets Interactively**
- Semantics of MLL Proofs
- Semantics: Soundness and Completeness
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

## From Proof Nets to Diagram Algebras

## Temperley-Lieb Algebra

## Characterizing Planarity

## Composition

## Functional

We can think of a proof structure (set of axiom links) as a **copy-cat strategy**, and a switching as a counter-strategy. A proof structure will be a proof-net if its interaction with every counter-strategy yields a correct result.

Hence we define (Girard 1988):

$$f \perp g \equiv fg \text{ is cyclic}$$

*i.e.*  $(fg)^k = 1$  where  $k$  is the cardinality of the underlying set (of literal occurrences), and this is true for no smaller value of  $k$ .

This condition is directly inspired by the **long trip condition**, the earlier version of the proof net correctness condition used by (Girard 1987).

We can then define

$$S^\perp = \{g \mid \forall f \in S. f \perp g\}.$$

# Semantics of MLL Proofs

## Introduction

## Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- Understanding Proof Nets Interactively
- **Semantics of MLL Proofs**
- Semantics: Soundness and Completeness
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

## From Proof Nets to Diagram Algebras

## Temperley-Lieb Algebra

## Characterizing Planarity

## Composition

## Functional

We now give a semantics of MLL proofs by specifying, for each formula  $A$ , a set  $S$  of permutations on the set of literal occurrences  $|A|$ , such that  $S = S^{\perp\perp}$ .

For a literal, we specify the unique permutation (the identity).

$$\begin{aligned} S(A \otimes B) &= \{f + g \mid f \in S(A) \wedge g \in S(B)\}^{\perp\perp} \\ S(A \wp B) &= S(A^{\perp} \otimes B^{\perp})^{\perp}. \end{aligned}$$

Note that, for every formula  $A$ :  $S(A^{\perp}) = S(A)^{\perp}$ .

We extend this assignment to sequents  $\Gamma$  by treating  $\Gamma$  as the Par of its formulas.

# Semantics: Soundness and Completeness

Introduction

Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- Understanding Proof Nets Interactively
- Semantics of MLL Proofs
- **Semantics: Soundness and Completeness**
- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

From Proof Nets to Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional

Computation: The Copy-Cat Strategies and Information Flow  
Planar  $\lambda$ -calculus

**Proposition 3 (Semantic Soundness)** *If  $f$  is the permutation assigned to a sequent proof of  $\Gamma$ , then  $f \in S(\Gamma)$ .*

# Semantics: Soundness and Completeness

Introduction

Logic

- Multiplicative Proof Structures
- From Proof Nets to Semantics
- Assignment of Permutations to Sequent Proofs
- MLL Proof Nets
- Results on Proof Nets
- Understanding Proof Nets Interactively
- Semantics of MLL Proofs

• **Semantics: Soundness and Completeness**

- Cut-Elimination by Permutations
- Cut-Elimination by Permutations Ctd

From Proof Nets to Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional

Computation: The Copy-Cat Strategies and Information Flow  
Planar  $\lambda$ -calculus

**Proposition 5 (Semantic Soundness)** *If  $f$  is the permutation assigned to a sequent proof of  $\Gamma$ , then  $f \in S(\Gamma)$ .*

**Theorem 6 (Full Completeness)** *If  $f \in S(\Gamma)$  is a literal-respecting involution, then  $f$  is a proof-net, and hence is the denotation of a sequent proof.*

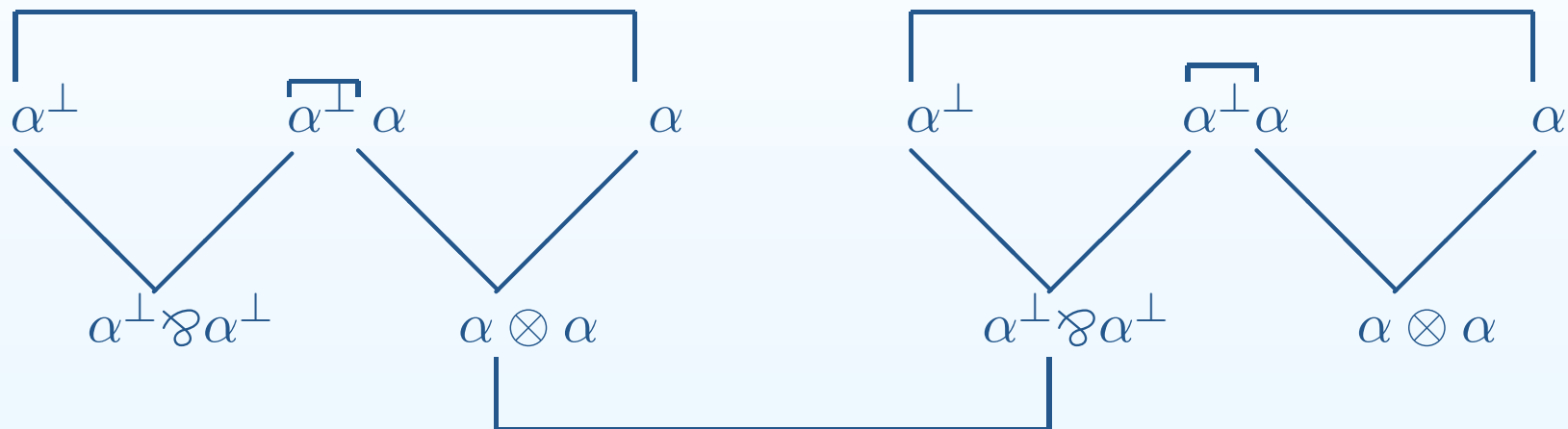
This shows **the equivalence of the geometric and interactive criteria for proofs.**

**Proof Outline:** Given  $\sigma \in S(\Gamma)$ , we assume that for some switching  $S$ ,  $G(\Gamma, \sigma, S)$  is not a tree. Then we construct a **counter-strategy**  $\tau \in S(\Gamma)^\perp$  such that  $\neg(\sigma \perp \tau)$ . Contradiction.

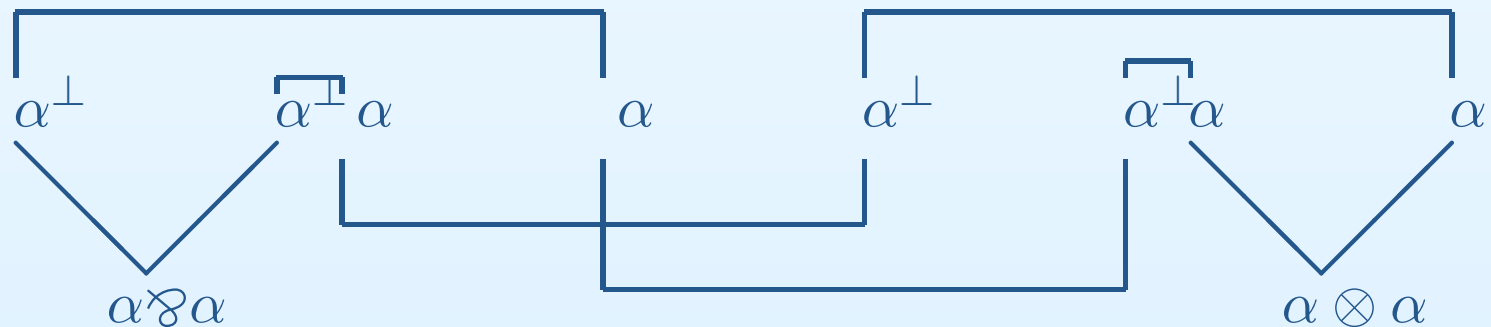
**Discussion:** Non-emptiness of  $S(A)$ , “paraproofs”, and uniformity.

## Cut-Elimination by Permutations

We consider performing Cut-elimination on  $\text{twist} \circ \text{twist}$ : The proof net for  $\text{twist} \circ \text{twist}$  before cut elimination is:

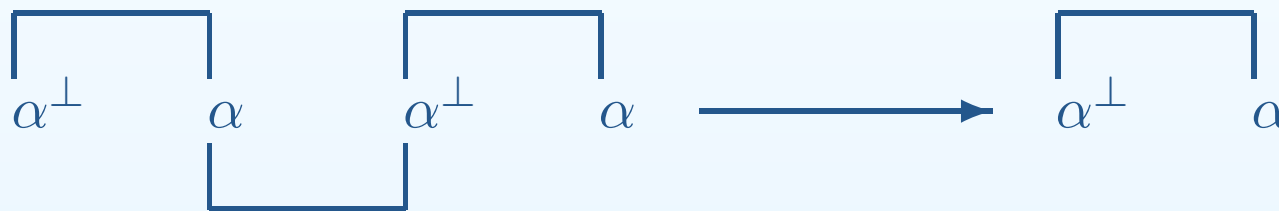


The proof net for  $\text{twist} \circ \text{twist}$  after one step of Cut elimination is:

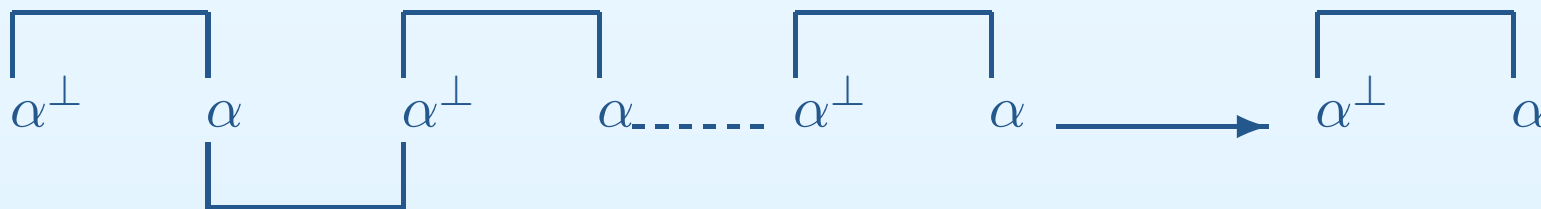


## Cut-Elimination by Permutations Ctd

Generally, in this fragment, we can apply this ‘decomposition rule’ repeatedly for tensors cut against par until all cuts are between axiom links. We can say that the whole purpose of these transformations is to match up the corresponding axiom links correctly; the ‘real’ information flow is then accomplished by the axiom reductions:



or more generally,



Two views: **geometric** and **interactive**.

Introduction

Logic

**From Proof Nets to  
Diagram Algebras**

- From Proof Nets to Diagram Algebras
- Diagrams for Arrows
- Example

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# From Proof Nets to Diagram Algebras

## From Proof Nets to Diagram Algebras

We now make a transition from an apparently very specialized corner of Proof Theory to a broad topic arising in Representation Theory, Knot Theory, and with connections to Mathematical Physics.

We shall on the one hand **lose** some structure, and on the other **gain** some.

- We shall obliterate the distinction between  $\otimes$  and  $\wp$ ; this corresponds to moving from **\*-autonomous** to **compact closed** categories. This means that we can forget about the formula tree structure altogether; we are simply connecting up literal occurrences, which we shall draw as “joining up the dots”.

Motivation: compact closed categories show up in many contexts of interest!

## Diagrams for Arrows

- On the other hand, rather than one-sided sequents, we shall represent general arrows or two-sided sequents diagrammatically. This means we represent arrows

$$A_1 \otimes \cdots \otimes A_n \longrightarrow B_1 \otimes \cdots \otimes B_m$$

where each  $A_i$  and  $B_j$  is a literal. We represent such arrows by literal-preserving involutions on  $\{1, \dots, n\} + \{1, \dots, m\}$ , where literal-preserving now means:

- We connect **opposite** literals in the domain or codomain, or
- We connect occurrences of the **same** literal in the domain and the codomain.

An advantage of this representation is that we express composition very transparently, by “stacking” arrows.

# Example

Introduction

Logic

From Proof Nets to Diagram Algebras

- From Proof Nets to Diagram Algebras
- Diagrams for Arrows
- Example

Temperley-Lieb Algebra

Characterizing Planarity

Composition

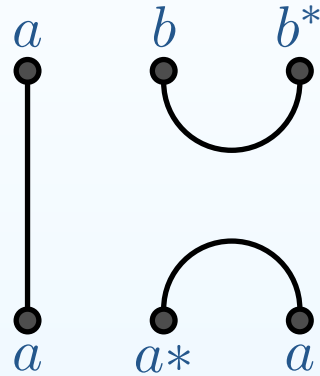
Functional Computation: The Planar  $\lambda$ -calculus

Functional Computation: Non-Planar Combinators

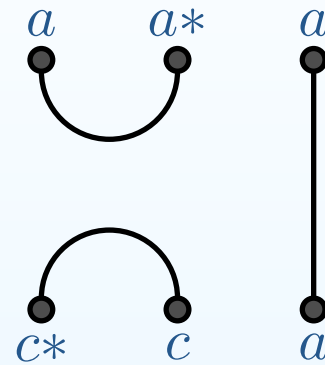
Logic of Quantum Information Flow

Cloning

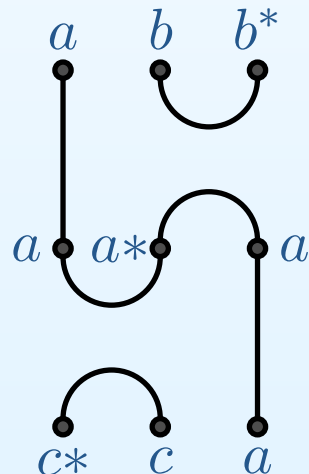
The composition of



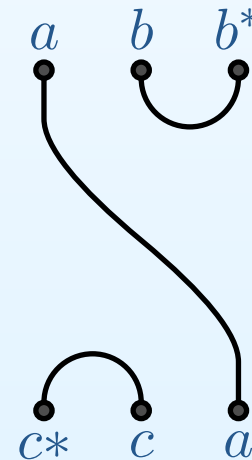
and



is given by



=



Introduction

Logic

From Proof Nets to  
Diagram Algebras

**Temperley-Lieb Algebra**

- Temperley-Lieb Algebra
- TL algebra: generators and relations
- Temperley-Lieb Monoids
- Diagram Monoids: Generators
- Diagram Monoids: Relations
- Expressiveness of the Generators
- Nested Cups and Caps
- The Trace
- Trace of Identity is the Dimension
- The Connection to Knots
- An Algebraic View

Characterizing Planarity

Composition

Functional

# Temperley-Lieb Algebra

# Temperley-Lieb Algebra

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

● **Temperley-Lieb  
Algebra**

● TL algebra:  
generators and relations

● Temperley-Lieb  
Monoids

● Diagram Monoids:  
Generators

● Diagram Monoids:  
Relations

● Expressiveness of the  
Generators

● Nested Cups and  
Caps

● The Trace

● Trace of Identity is the  
Dimension

● The Connection to  
Knots

● An Algebraic View

Characterizing Planarity

Composition

Functional

Computation: The  
Copy-Cat Strategies and Information Flow  
Planar  $\lambda$ -calculus

The **Temperley-Lieb algebra** played a central role in the **Jones polynomial invariant of knots** and ensuing developments.

The TL algebra was originally presented, rather forbiddingly, in terms of abstract generators and relations. It was recast in beautifully elementary and conceptual terms by Louis Kauffman as a **planar diagram algebra**.

## TL algebra: generators and relations

We fix a ring  $R$ . Given a choice of **parameter**  $\tau \in R$  and a **dimension**  $n \in \mathbb{N}$ , we define the Temperley-Lieb algebra  $\mathcal{A}_n(\tau)$  to be the unital, associative  $R$ -linear algebra with generators

$$U_1, \dots, U_{n-1}$$

and relations

$$\begin{aligned} U_i U_j U_i &= U_i & |i - j| = 1 \\ U_i^2 &= \tau \cdot U_i \\ U_i U_j &= U_j U_i & |i - j| > 1 \end{aligned}$$

Note that the only relations used in defining the algebra are multiplicative ones. This suggests that we can present the multiplicative monoid  $\mathcal{M}_n$ , and then obtaining  $\mathcal{A}_n(\tau)$  as the **monoid algebra** of formal  $R$ -linear combinations  $\sum_i r_i \cdot a_i$  over  $\mathcal{M}_n$ , with multiplication defined by bilinear extension:

$$\left( \sum_i r_i \cdot a_i \right) \left( \sum_j s_j \cdot b_j \right) = \sum_{i,j} (r_i s_j) \cdot (a_i b_j).$$

# Temperley-Lieb Monoids

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

- Temperley-Lieb Algebra
- TL algebra: generators and relations
- **Temperley-Lieb Monoids**
- Diagram Monoids: Generators
- Diagram Monoids: Relations
- Expressiveness of the Generators
- Nested Cups and Caps
- The Trace
- Trace of Identity is the Dimension
- The Connection to Knots
- An Algebraic View

Characterizing Planarity

Composition

Functional

Computation: The  
Copy-Cat Strategies and Information Flow  
Planar  $\lambda$ -calculus

We define  $\mathcal{M}_n$  as the monoid with generators

$$\delta, U_1, \dots, U_{n-1}$$

and relations

$$\begin{aligned} U_i U_j U_i &= U_i & |i - j| = 1 \\ U_i^2 &= \delta U_i \\ U_i U_j &= U_j U_i & |i - j| > 1 \\ \delta U_i &= U_i \delta \end{aligned}$$

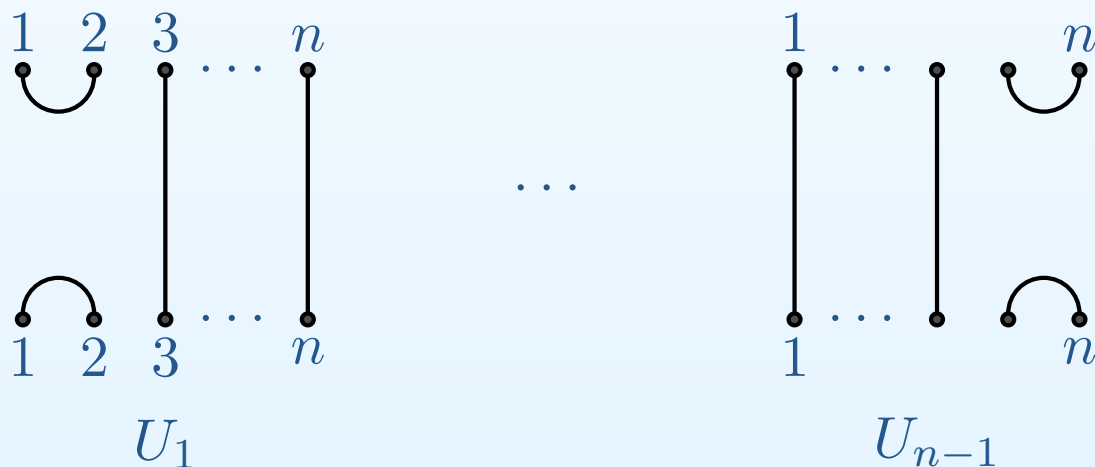
We can then obtain  $\mathcal{A}_n(\tau)$  as the monoid algebra over  $\mathcal{M}_n$ , subject to the identification

$$\delta = \tau \cdot 1.$$

## Diagram Monoids: Generators

We start with two parallel rows of  $n$  dots (geometrically, points in the plane). An element of the monoid is obtained by “joining up the dots” pairwise in a smooth, planar fashion, where the arc connecting each pair of dots must lie within the rectangle framing the two parallel rows of dots. Such diagrams are identified up to planar isotopy, *i.e.* continuous deformation within the portion of the plane bounded by the framing rectangle..

The generators  $U_1, \dots, U_{n-1}$  can be drawn as follows:

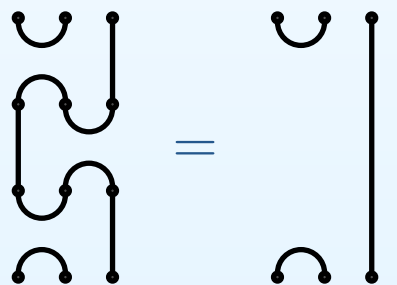


The generator  $\delta$  corresponds to a loop  $\bigcirc$  — all such loops are identified up to isotopy.

## Diagram Monoids: Relations

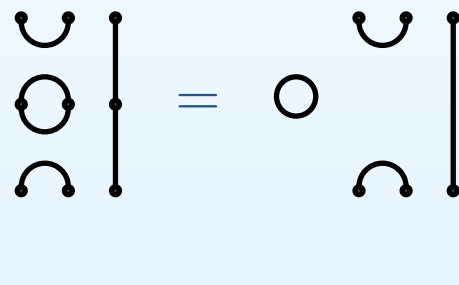
We refer to arcs connecting dots in the top row as **cups**, those connecting dots in the bottom row as **caps**, and those connecting a dot in the top row to a dot in the bottom row as **through lines**.

Multiplication  $xy$  is defined by identifying the bottom row of  $x$  with the top row of  $y$ , and composing paths. In general loops may be formed — these are “scalars”, which can float freely across these figures. The relations can be illustrated as follows:



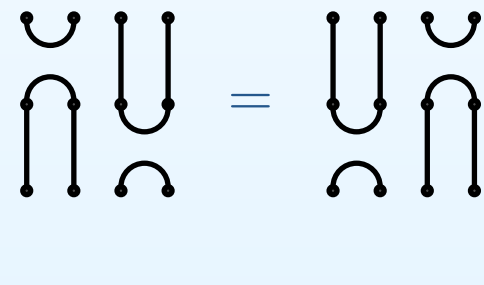
The diagram shows two vertical lines. The left line has a cup at the top, a through line, a cap at the bottom, a through line, a cup at the top, a through line, a cap at the bottom, a through line, and a cap at the bottom. The right line has a cup at the top, a through line, a cap at the bottom, a through line, and a cap at the bottom. An equals sign is between them.

$$U_1 U_2 U_1 = U_1$$



The diagram shows a vertical line with a cup at the top, a loop (a circle with a dot in the center) in the middle, and a cap at the bottom. This is equal to a scalar (a circle with a dot in the center) multiplied by a vertical line with a cup at the top and a cap at the bottom.

$$U_1^2 = \delta U_1$$

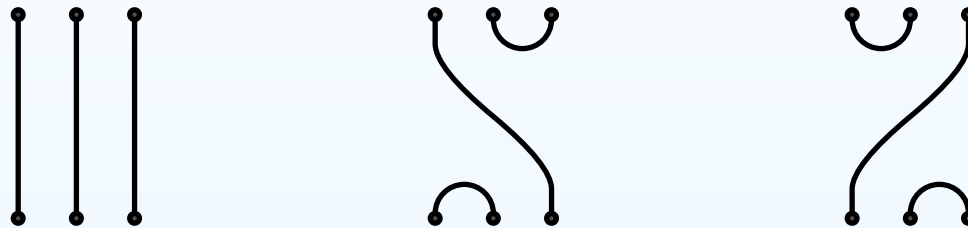


The diagram shows two vertical lines. The left line has a cup at the top, a through line, a cap at the bottom, a through line, a cup at the top, and a through line. The right line has a cup at the top, a through line, a cap at the bottom, a through line, a cup at the top, and a through line. An equals sign is between them.

$$U_1 U_3 = U_3 U_1$$

## Expressiveness of the Generators

The fact that all planar diagrams can be expressed as products of generators is not entirely obvious. As an illustrative example, consider the planar diagrams in  $\mathcal{M}_3$ . Apart from the generators  $U_1, U_2$ , and ignoring loops, there are three:



The first is the identity for the monoid; we refer to the other two as the **left wave** and **right wave** respectively. The left wave can be expressed as the product  $U_2U_1$ :



The right wave has a similar expression.

## Nested Cups and Caps

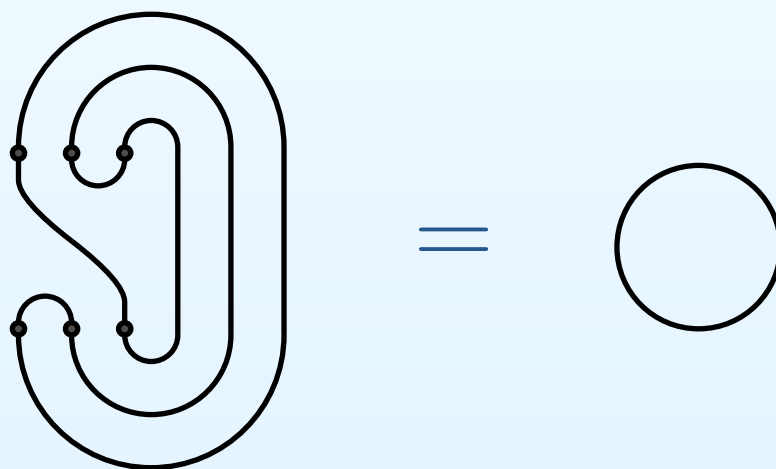
Once we are in dimension four or higher, we can have **nested cups and caps**. These can be built using waves, as illustrated by the following:



## The Trace

There is a natural **trace function** on the Temperley-Lieb algebra, which can be defined diagrammatically on  $\mathcal{M}_n$  by connecting each dot in the top row to the corresponding dot in the bottom row, using auxiliary cups and caps. This always yields a diagram isotopic to a number of loops — hence to a **scalar**, as expected. This trace can then be extended linearly to  $\mathcal{A}_n(\tau)$ .

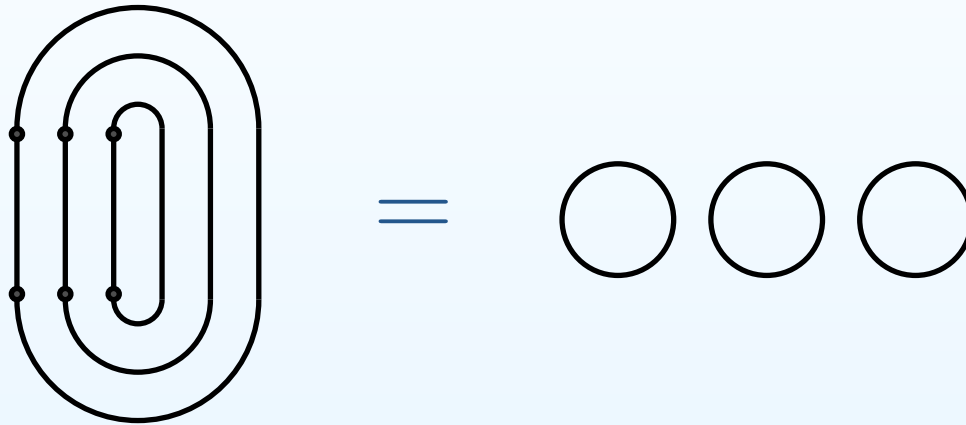
We illustrate this firstly by taking the trace of a wave—which is equal to a single loop:



The Ear is a Circle

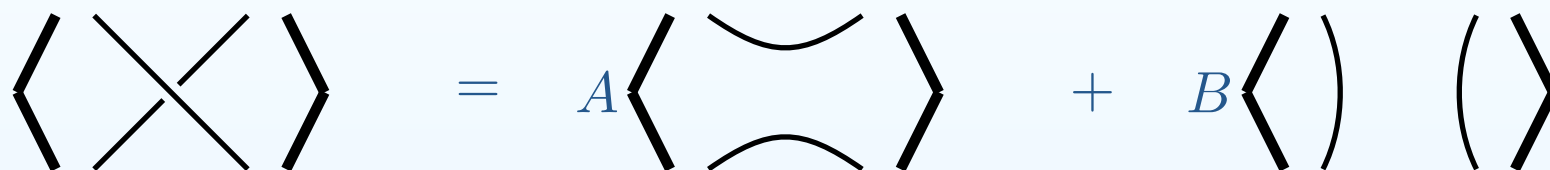
## Trace of Identity is the Dimension

Our second example illustrates the important general point that **the trace of the identity in  $\mathcal{M}_n$  is  $\delta^n$** :



## The Connection to Knots

How does this connect to knots? Again, a key conceptual insight is due to Kauffman, who saw how to recast the Jones polynomial in elementary combinatorial form in terms of his **bracket polynomial**. The basic idea of the bracket polynomial is expressed by the following equation:

$$\langle \text{crossing} \rangle = A \langle \text{smoothing 1} \rangle + B \langle \text{smoothing 2} \rangle$$
The diagram shows an equation between three terms. On the left is a crossing of two strands, each enclosed in a thick black bracket. This is equal to the sum of two terms. The first term is coefficient A multiplied by a smoothing where the two strands are connected by two arcs, also enclosed in thick black brackets. The second term is coefficient B multiplied by two separate arcs, each enclosed in a thick black bracket.

Each over-crossing in a knot or link is evaluated to a weighted sum of the two possible planar smoothings. With suitable choices for the coefficients  $A$  and  $B$  (as Laurent polynomials), this is invariant under the second and third Reidemeister moves. With an ingenious choice of normalizing factor, it becomes invariant under the first Reidemeister move — and yields the Jones polynomial!

## An Algebraic View

What this means algebraically is that the braid group  $\mathcal{B}_n$  has a representation in the Temperley-Lieb algebra  $\mathcal{A}_n(\tau)$  — the above bracket evaluation shows how the generators  $\beta_i$  of the braid group are mapped into the Temperley-Lieb algebra:

$$\beta_i \mapsto A \cdot U_i + B \cdot 1.$$

Every knot arises as the closure (*i.e.* the diagrammatic trace) of a braid; the invariant arises by mapping the **open braid** into the Temperley-Lieb algebra, and taking the trace there.

This is just the beginning of a huge swathe of further developments, including: Topological Quantum Field Theories, Quantum Groups, Quantum Statistical mechanics, Diagram Algebras and Representation Theory, and more.

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

**Characterizing Planarity**

- Characterizing Planarity
- First condition
- Second condition
- Planarity for Points
- Names and Conames
- Example

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# Characterizing Planarity

# Characterizing Planarity

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

● Characterizing  
Planarity

- First condition
- Second condition
- Planarity for Points
- Names and Conames
- Example

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

A map  $f \in \text{Inv}(N(n, m))$  will be called **planar** if it satisfies the following two conditions, for all  $i, j \in N(n, m)$ :

$$\text{(PL1)} \quad i < j < f(i) \quad \Longrightarrow \quad f(j) < f(i)$$

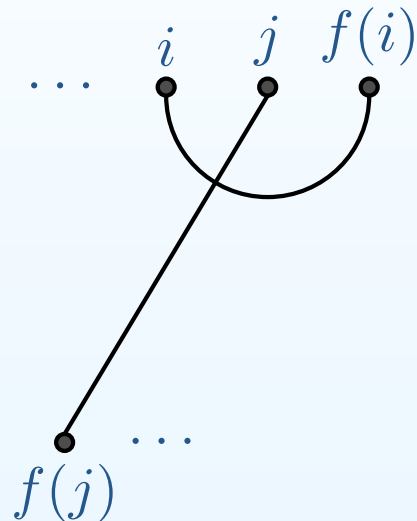
$$\text{(PL2)} \quad f(i) \# i < j \# f(j) \quad \Longrightarrow \quad f(i) < f(j).$$

It is instructive to see which possibilities are **excluded** by these conditions.

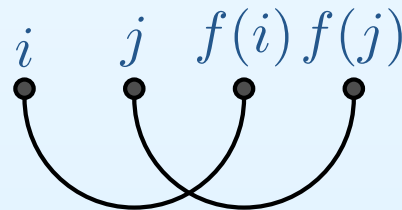
# First condition

$$(PL1) \quad i < j < f(i) \implies f(j) < f(i)$$

(PL1) rules out



where  $f(j) \neq f(i)$ , and also



where  $f(i) < f(j)$ .

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

● Characterizing  
Planarity

● **First condition**

● Second condition

● Planarity for Points

● Names and Conames

● Example

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

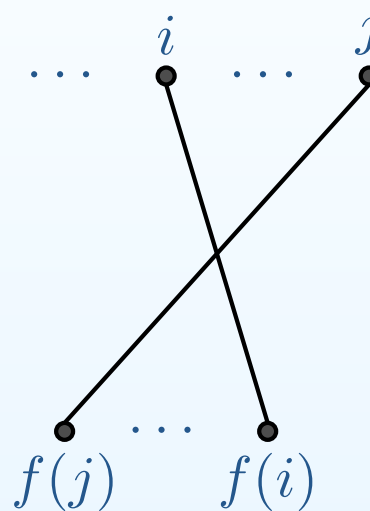
Logic of Quantum  
Information Flow

Cloning

## Second condition

$$(PL2) \quad f(i) \# i < j \# f(j) \implies f(i) < f(j).$$

Similarly, (PL2) rules out



We write  $\mathcal{P}(n, m)$  for the set of planar maps in  $\text{Inv}(\mathbf{N}(n, m))$ .

### Proposition 7

1. *Every planar diagram satisfies the two conditions.*
2. *Every involution satisfying the two conditions can be drawn as a planar diagram.*

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

- Characterizing Planarity
- First condition
- **Second condition**
- Planarity for Points
- Names and Conames
- Example

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

## Planarity for Points

Rather than proving this directly, it is simpler, and also instructive, to reduce it to a special case. We consider arrows in  $\mathcal{D}$  of the special form  $I \rightarrow \mathbf{n}$ . Such arrows consist only of caps. They correspond to **points**, or **states** in the terminology of Categorical QM.

Since the top row of dots is empty, in this case we have a linear order, and the premise of condition (PL2) can never arise. Hence planarity for such arrows is just the simple condition (PL1) — which can be seen to be equivalent to saying that, if we write a left parenthesis for each left end of a cap, and a right parenthesis for each right end, we get a well-formed string of parentheses.

Thus



corresponds to

$()(())$ .

## Names and Conames

Now we recall that quite generally, in any pivotal category we have the Hom-Tensor adjunction

$$A \otimes B^* \xrightarrow{[f]} I \xleftarrow{\cong} A \xrightarrow{f} B \xleftarrow{\cong} I \xrightarrow{[f]} A^* \otimes B$$

$$\lceil f \rceil = (1_{A^*} \otimes f) \circ \eta_A : I \rightarrow A^* \otimes B \quad \lfloor f \rfloor = \epsilon_B \circ (f \otimes 1_{B^*}) : A \otimes B^* \rightarrow I.$$

We call  $\lceil f \rceil$  the **name** of  $f$ , and  $\lfloor f \rfloor$  the **coname**. The inverse to the map  $f \mapsto \lceil f \rceil$  is defined by

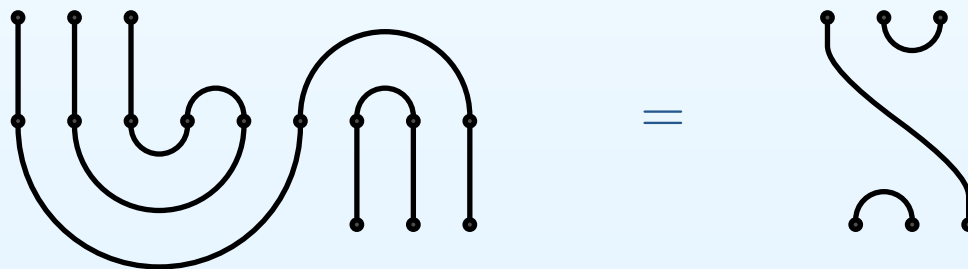
$$g : I \rightarrow A^* \otimes B \mapsto (\epsilon_A \otimes 1_B) \circ (1_A \otimes g) : A \rightarrow B.$$

## Example

We compute the name of the left wave:



Applying the inverse transformation:



Note also that **the unit is the name of the identity**:  $\eta_{\mathbf{n}} = \lceil 1_{\mathbf{n}} \rceil$ , and similarly  $\epsilon_{\mathbf{n}} = \lfloor 1_{\mathbf{n}} \rfloor$ .

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

**Composition**

- The Temperley-Lieb Category
- Composition
- The 'Execution Formula'
- Reading the Execution Formula
- Reading the Execution Formula

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# Composition

# The Temperley-Lieb Category

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

● The Temperley-Lieb  
Category

- Composition
- The 'Execution  
Formula'
- Reading the  
Execution Formula
- Reading the  
Execution Formula

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

Our aim is now to define a category  $\mathcal{T}$ , which will yield the desired description of the Temperley-Lieb monoids. The **objects** of  $\mathcal{T}$  are the natural numbers. The homset  $\mathcal{T}(n, m)$  is defined to be the cartesian product  $\mathbb{N} \times \mathcal{P}(n, m)$ . Thus a morphism  $n \rightarrow m$  in  $\mathcal{T}$  consists of a pair  $(s, f)$ , where  $s$  is a natural number, and  $f \in \mathcal{P}(n, m)$  is a planar map in  $\text{Inv}(\mathbb{N}(n, m))$ .

It remains to define the composition and identities in this category. Clearly (even leaving aside the natural number components of morphisms) composition cannot be defined as ordinary function composition. This does not even make sense — the codomain of a morphism  $f : n \rightarrow m$  does not match the domain of a morphism  $g : m \rightarrow p$  — let alone yield a function with the necessary properties to be a morphism in the category.

# Composition

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

- The Temperley-Lieb  
Category

- **Composition**

- The 'Execution  
Formula'
- Reading the  
Execution Formula
- Reading the  
Execution Formula

Functional

Computation: The  
Planar  $\lambda$ -calculus

Functional

Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

Consider a map  $f : [n] + [m] \longrightarrow [n] + [m]$ . Each input lies in **either**  $[n]$  **or**  $[m]$  (exclusive or), and similarly for the corresponding output. This leads to a decomposition of  $f$  into four **disjoint partial maps**:

$$\begin{array}{ll} f_{n,n} : [n] \longrightarrow [n] & f_{n,m} : [n] \longrightarrow [m] \\ f_{m,n} : [m] \longrightarrow [n] & f_{m,m} : [m] \longrightarrow [m] \end{array}$$

so that  $f$  can be recovered as the disjoint union of these four maps. If  $f$  is an involution, then these maps will be partial involutions.

Now suppose we have maps  $f : [n] + [m] \rightarrow [n] + [m]$  and  $g : [m] + [p] \rightarrow [m] + [p]$ . We write the decompositions of  $f$  and  $g$  as above in matrix form:

$$f = \begin{pmatrix} f_{n,n} & f_{n,m} \\ f_{m,n} & f_{m,m} \end{pmatrix} \quad g = \begin{pmatrix} g_{m,m} & g_{m,p} \\ g_{p,m} & g_{p,p} \end{pmatrix}$$

## The ‘Execution Formula’

We can view these maps as **binary relations** on  $[n] + [m]$  and  $[m] + [p]$  respectively, and use relational algebra (union  $R \cup S$ , relational composition  $R; S$  and reflexive transitive closure  $R^*$ ) to define a **new relation**  $\theta$  on  $[n] + [p]$ . If we write

$$\theta = \begin{pmatrix} \theta_{n,n} & \theta_{n,p} \\ \theta_{p,n} & \theta_{p,p} \end{pmatrix}$$

so that  $\theta$  is the disjoint union of these four components, then we can define it component-wise as follows:

$$\begin{aligned} \theta_{n,n} &= f_{n,n} \cup f_{n,m}; g_{m,m}; (f_{m,m}; g_{m,m})^*; f_{m,n} \\ \theta_{n,p} &= f_{n,m}; (g_{m,m}; f_{m,m})^*; g_{m,p} \\ \theta_{p,n} &= g_{p,m}; (f_{m,m}; g_{m,m})^*; f_{m,n} \\ \theta_{p,p} &= g_{p,p} \cup g_{p,m}; f_{m,m}; (g_{m,m}; f_{m,m})^*; g_{m,p}. \end{aligned}$$

## Reading the Execution Formula

We can give clear intuitive readings for how these formulas express composition of paths in diagrams in terms of relational algebra:

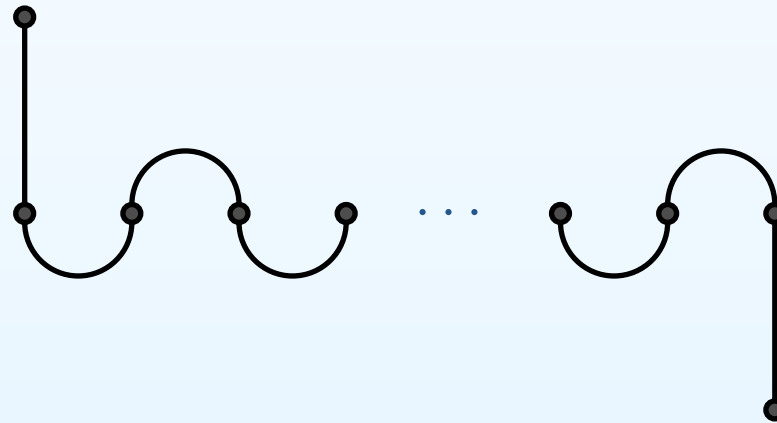
- The component  $\theta_{n,n}$  describes the **cups** of the diagram resulting from the composition. These are the union of the cups of  $f$  ( $f_{n,n}$ ), together with paths that start from the top row with a through line of  $f$ , given by  $f_{n,m}$ , then go through an alternating odd-length sequence of cups of  $g$  ( $g_{m,m}$ ) and caps of  $f$  ( $f_{m,m}$ ), and finally return to the top row by a through line of  $f$  ( $f_{m,n}$ ).



$$f_{n,m}; g_{m,m}; (f_{m,m}; g_{m,m})^*; f_{m,n}$$

## Reading the Execution Formula

- Similarly,  $\theta_{p,p}$  describes the caps of the composition.
- $\theta_{n,p} = \theta_{p,n}^c$  describe the through lines. Thus  $\theta_{n,p}$  describes paths which start with a through line of  $f$  from  $n$  to  $m$ , continue with an alternating even-length (and possibly empty) sequence of cups of  $g$  and caps of  $f$ , and finish with a through line of  $g$  from  $m$  to  $p$ .



$$f_{n,m}; (g_{m,m}; f_{m,m})^*; g_{m,p}$$

All through lines from  $n$  to  $p$  must have this form.

**Proposition 8** *If  $f$  and  $g$  are planar, so is  $\theta$ .*

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

**Functional  
Computation: The  
Planar  $\lambda$ -calculus**

- An Example
- Example ctd

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

Cloning

# Functional Computation: The Planar $\lambda$ -calculus

## An Example

We shall consider the **bracketing combinator**

$$\mathbf{B} \equiv \lambda x. \lambda y. \lambda z. x(yz) : (B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C).$$

This is characterized by the equation  $\mathbf{B}abc = a(bc)$ .

## An Example

We shall consider the **bracketing combinator**

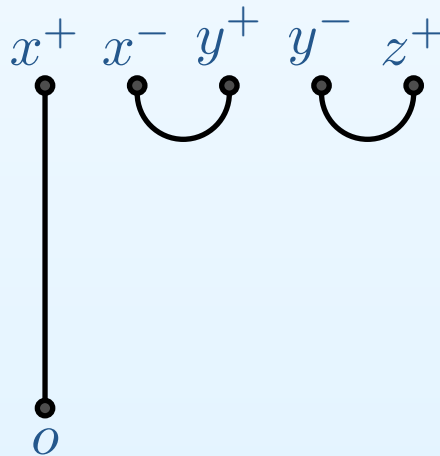
$$\mathbf{B} \equiv \lambda x. \lambda y. \lambda z. x(yz) : (B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C).$$

This is characterized by the equation  $\mathbf{B}abc = a(bc)$ .

We take  $A = B = C = \mathbf{1}$  in  $\mathbf{TL}$ . The interpretation of the open term

$$x : B \rightarrow C, y : A \rightarrow B, z : A \vdash x(yz) : C$$

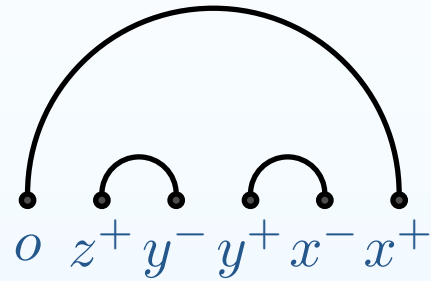
is as follows:



Here  $x^+$  is the output of  $x$ , and  $x^-$  the input, and similarly for  $y$ . The output of the whole expression is  $o$ .

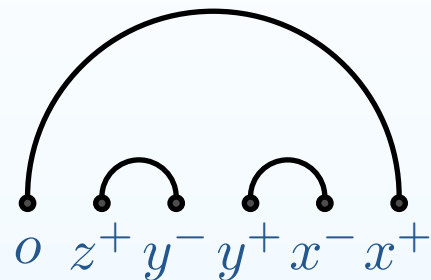
## Example ctd

When we abstract the variables, we obtain the following caps-only diagram:

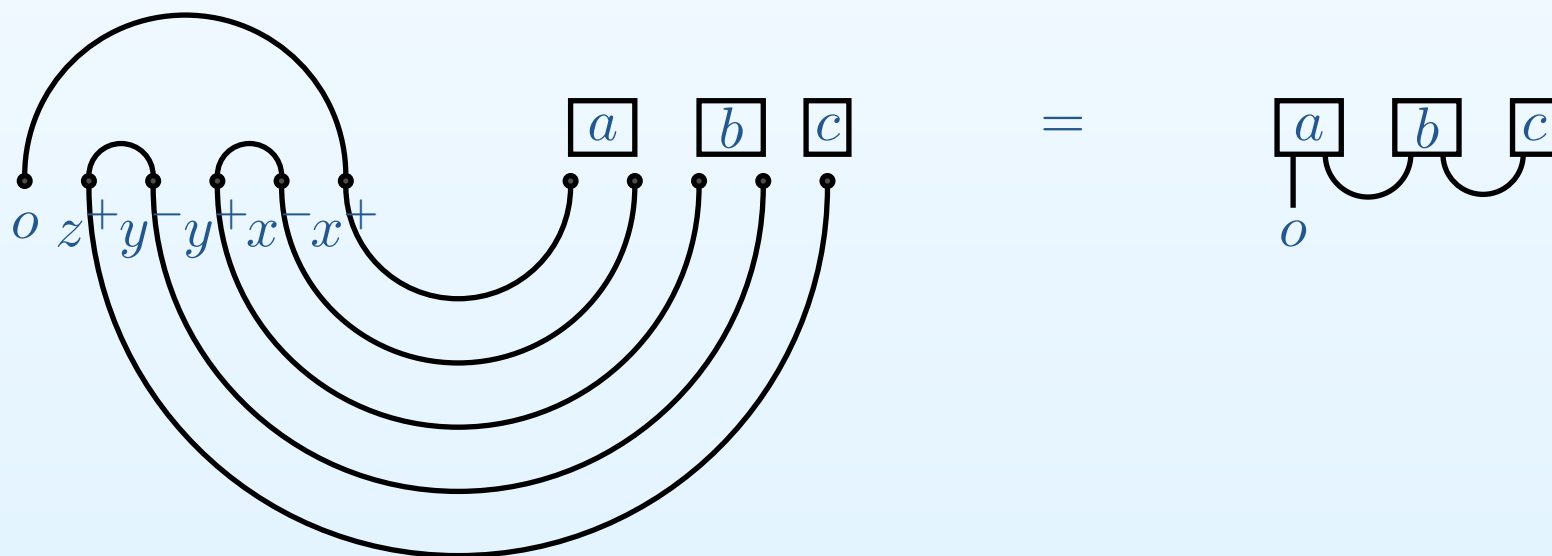


## Example ctd

When we abstract the variables, we obtain the following caps-only diagram:



Now we consider an application  $\mathbf{B}abc$ :



Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

**Functional  
Computation:  
Non-Planar  
Combinators**

- An Example
- Example ctd
- Wider perspective

Logic of Quantum  
Information Flow

Cloning

# Functional Computation: Non-Planar Combinators

## An Example

We shall consider the **commuting combinator**

$$\mathbf{C} \equiv \lambda x. \lambda y. \lambda z. xzy : (A \rightarrow B \rightarrow C) \rightarrow B \rightarrow A \rightarrow C.$$

This is characterized by the equation  $\mathbf{C}abc = acb$ .

## An Example

We shall consider the **commuting combinator**

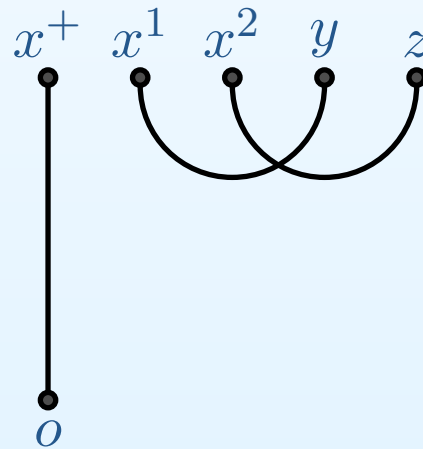
$$\mathbf{C} \equiv \lambda x. \lambda y. \lambda z. xzy : (A \rightarrow B \rightarrow C) \rightarrow B \rightarrow A \rightarrow C.$$

This is characterized by the equation  $\mathbf{C}abc = acb$ .

The interpretation of the open term

$$x : A \rightarrow B \rightarrow C, y : B, z : A \vdash xzy : C$$

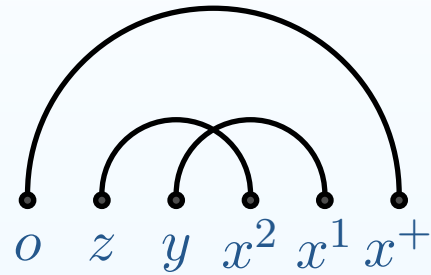
is as follows:



Here  $x^+$  is the output of  $x$ ,  $x^1$  the first input, and  $x^2$  the second input. The output of the whole expression is  $o$ .

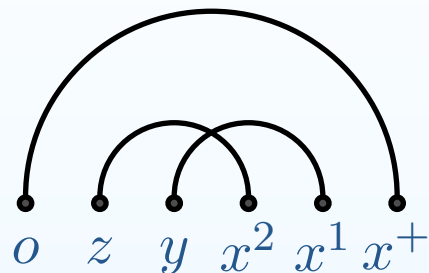
## Example ctd

When we abstract the variables, we obtain the following caps-only diagram:

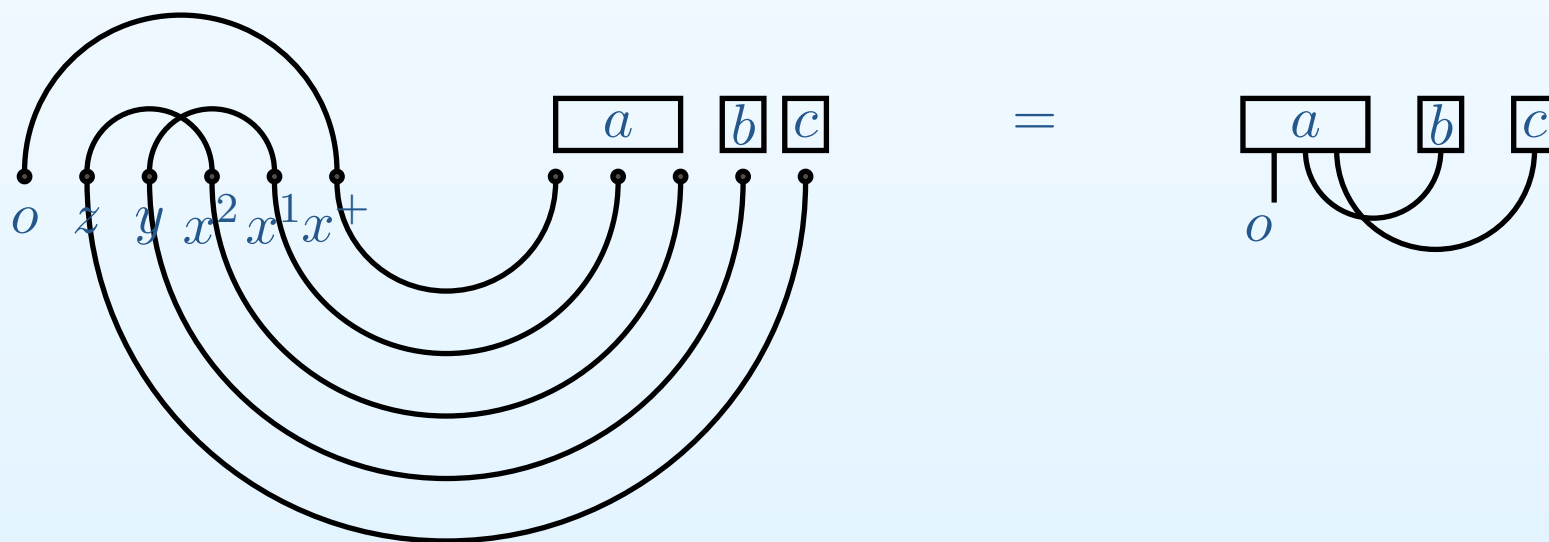


## Example ctd

When we abstract the variables, we obtain the following caps-only diagram:



Now we consider an application  $Cabc$ :



# Wider perspective

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

- An Example
- Example ctd
- **Wider perspective**

Logic of Quantum  
Information Flow

Cloning

## Wider perspective

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

- An Example
- Example ctd
- **Wider perspective**

Logic of Quantum  
Information Flow

Cloning

- The **Brauer algebra** (1931) arises if we drop the planarity condition on the TL algebra. This plays an important role in the representation theory of the Orthogonal group ('Schur-Weyl duality'). A whole genre of 'diagram algebras' in Representation Theory.

## Wider perspective

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

- An Example
- Example ctd
- **Wider perspective**

Logic of Quantum  
Information Flow

Cloning

- The **Brauer algebra** (1931) arises if we drop the planarity condition on the TL algebra. This plays an important role in the representation theory of the Orthogonal group ('Schur-Weyl duality'). A whole genre of 'diagram algebras' in Representation Theory.
- With **BCI** combinators one can interpret **Linear  $\lambda$ -calculus**.

## Wider perspective

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

- An Example
- Example ctd
- **Wider perspective**

Logic of Quantum  
Information Flow

Cloning

- The **Brauer algebra** (1931) arises if we drop the planarity condition on the TL algebra. This plays an important role in the representation theory of the Orthogonal group ('Schur-Weyl duality'). A whole genre of 'diagram algebras' in Representation Theory.
- With **BCI** combinators one can interpret **Linear  $\lambda$ -calculus**.
- One can retrieve the Kelly-Laplaza construction of the free compact closed category by a straightforward generalization.

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

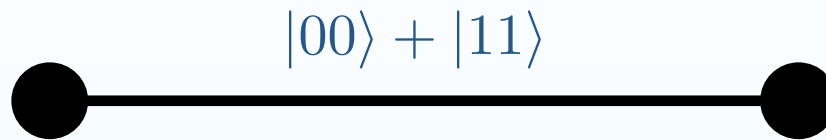
**Logic of Quantum  
Information Flow**

- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow
  - Projectors
- Decomposed
  - Compositionality

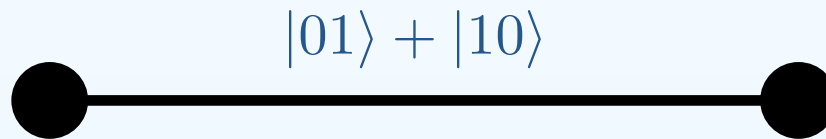
# Logic of Quantum Information Flow

# Quantum Entanglement

Bell state:

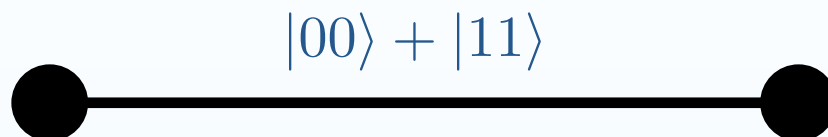


EPR state:

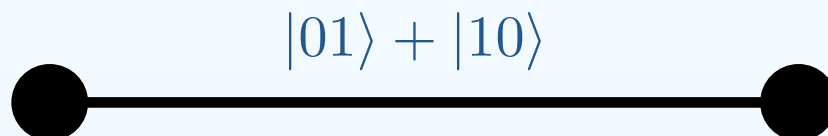


# Quantum Entanglement

Bell state:



EPR state:



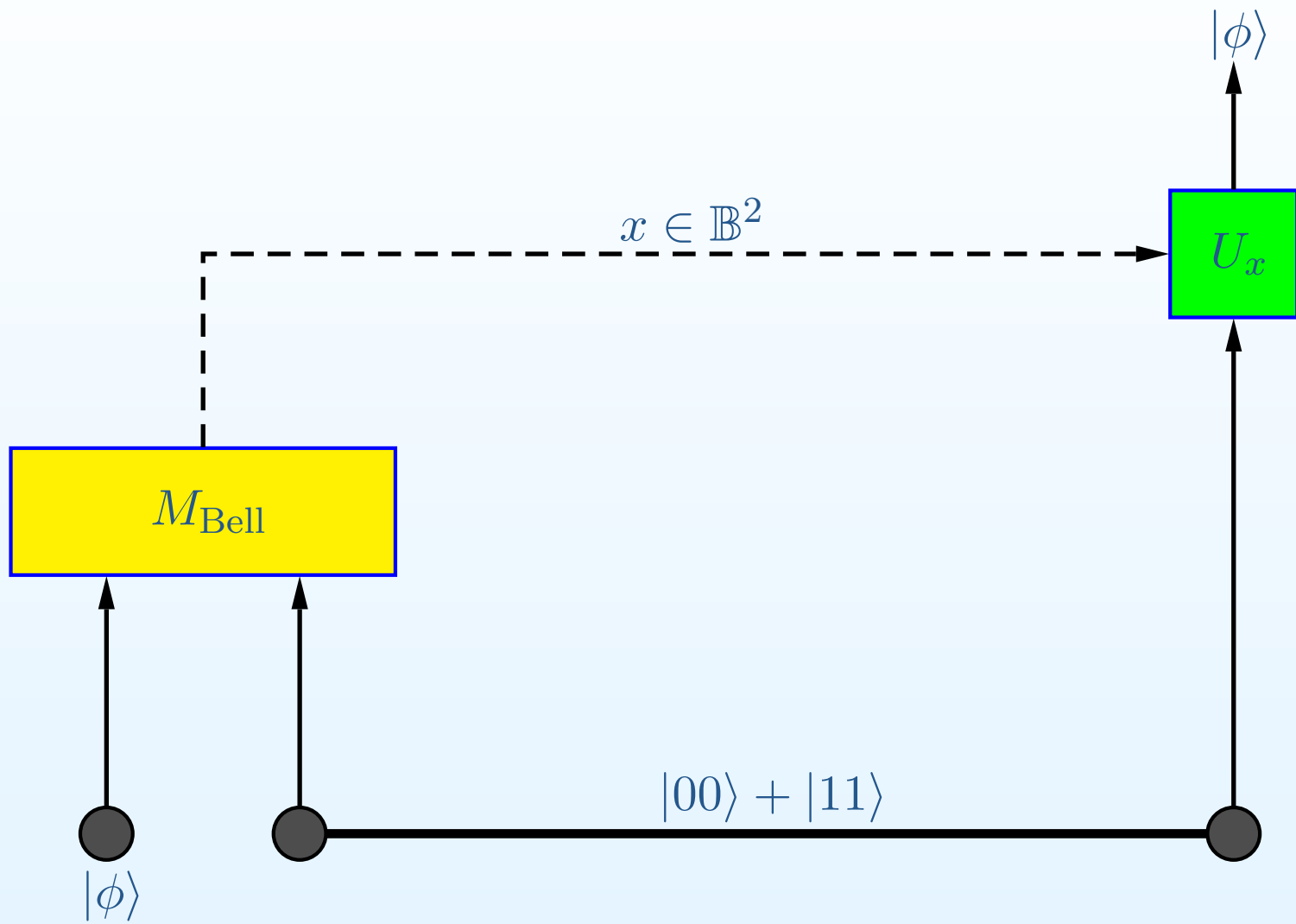
Compound systems are represented by **tensor product**:  $\mathcal{H}_1 \otimes \mathcal{H}_2$ . Typical element:

$$\sum_i \lambda_i \cdot \phi_i \otimes \psi_i$$

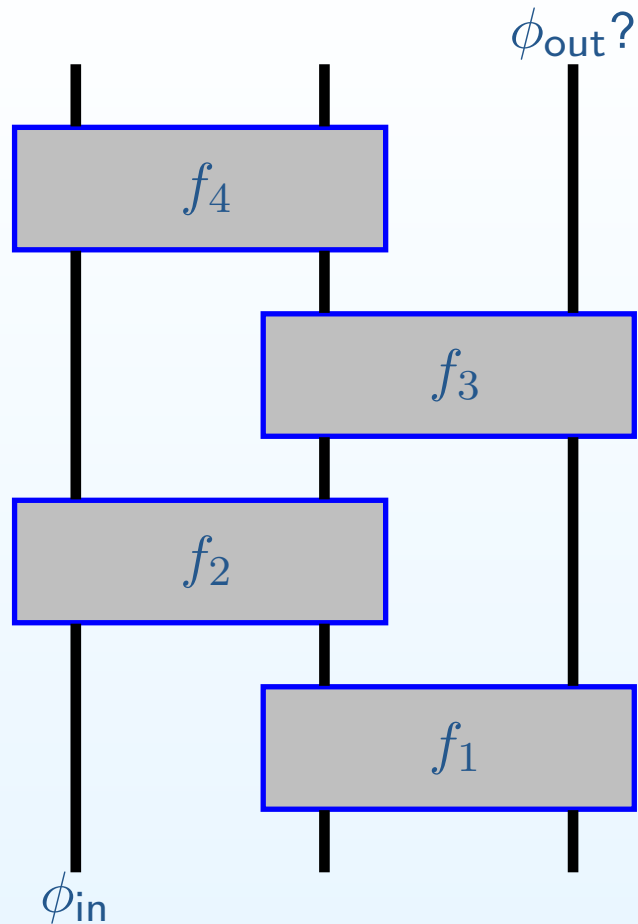
**Superposition** encodes **correlation**. Einstein's 'spooky action at a distance'. Even if the particles are spatially separated, measuring one has an effect on the state of the other.

Bell's theorem: QM is **essentially non-local**.

# From 'paradox' to 'feature': Teleportation

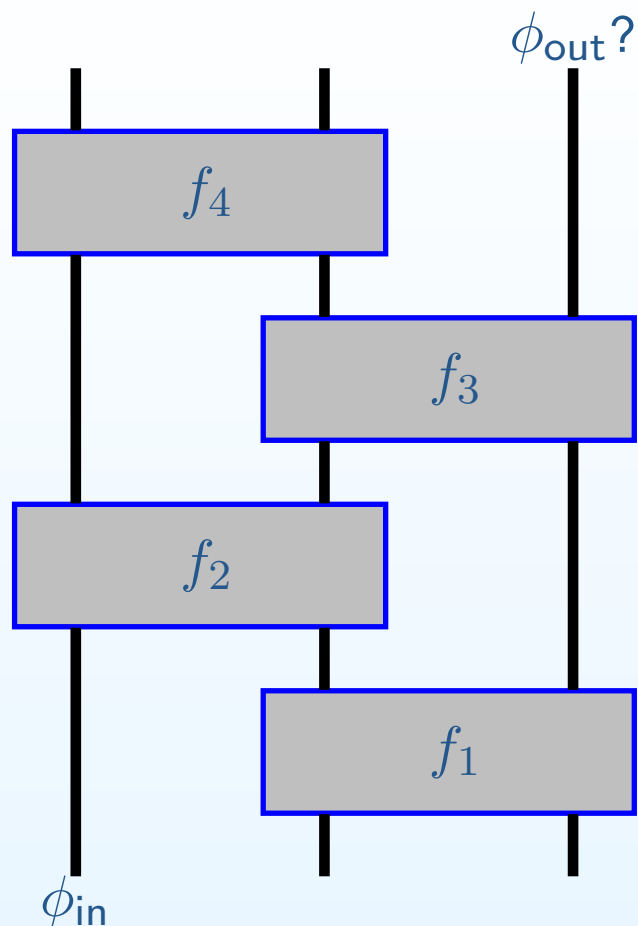


## What is the output?



$$(\mathcal{P}_{f_4} \otimes 1) \circ (1 \otimes \mathcal{P}_{f_3}) \circ (\mathcal{P}_{f_2} \otimes 1) \circ (1 \otimes \mathcal{P}_{f_1}) : \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3 \longrightarrow \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$$

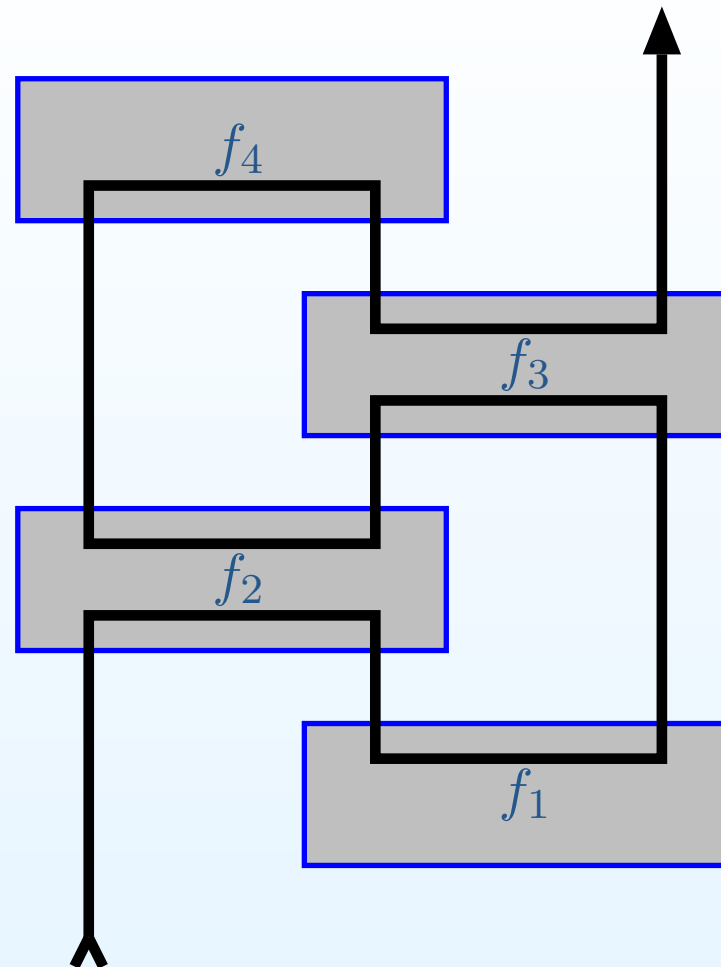
## What is the output?



$$(\mathcal{P}_{f_4} \otimes 1) \circ (1 \otimes \mathcal{P}_{f_3}) \circ (\mathcal{P}_{f_2} \otimes 1) \circ (1 \otimes \mathcal{P}_{f_1}) : \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3 \longrightarrow \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$$

$$\phi_{\text{out}} = f_3 \circ f_4 \circ f_2^\dagger \circ f_3^\dagger \circ f_1 \circ f_2(\phi_{\text{in}})$$

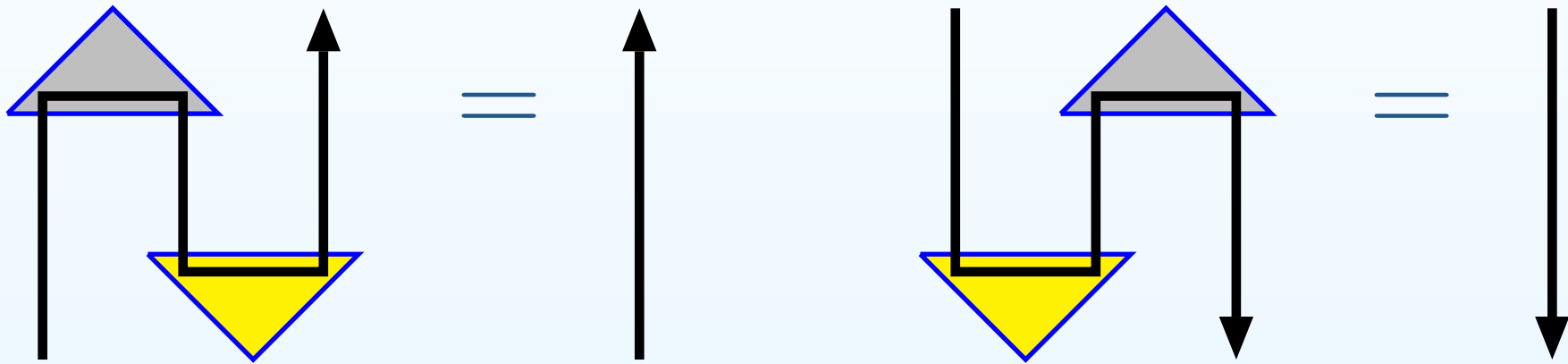
# Follow the line!



$$f_3 \circ f_4 \circ f_2^\dagger \circ f_3^\dagger \circ f_1 \circ f_2$$

# Graphical Calculus for Information Flow

**Compact Closure:** The basic algebraic laws for units and counits.



$$(\epsilon_A \otimes 1_A) \circ (1_A \otimes \eta_A) = 1_A$$

$$(1_{A^*} \otimes \epsilon_A) \circ (\eta_A \otimes 1_{A^*}) = 1_{A^*}$$

# Projectors Decomposed

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

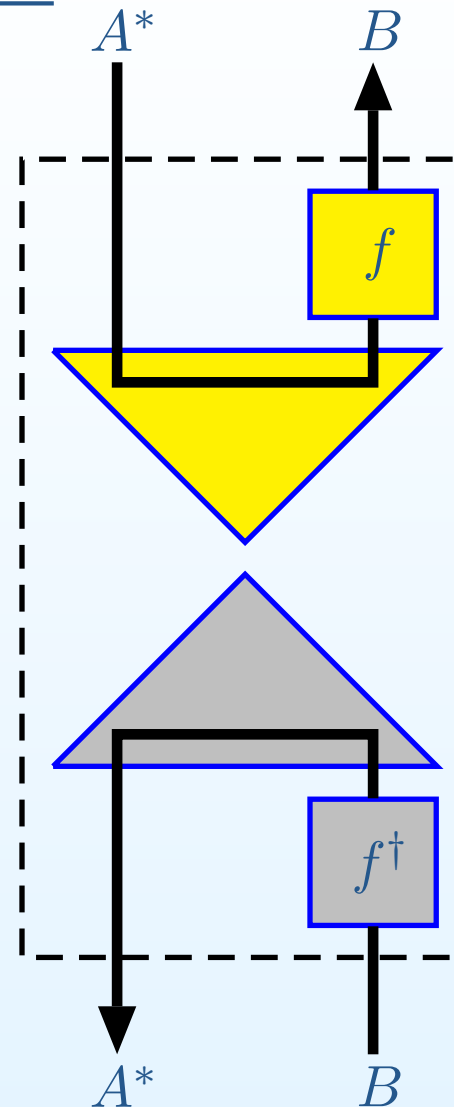
- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow

● **Projectors Decomposed**

● Compositionality

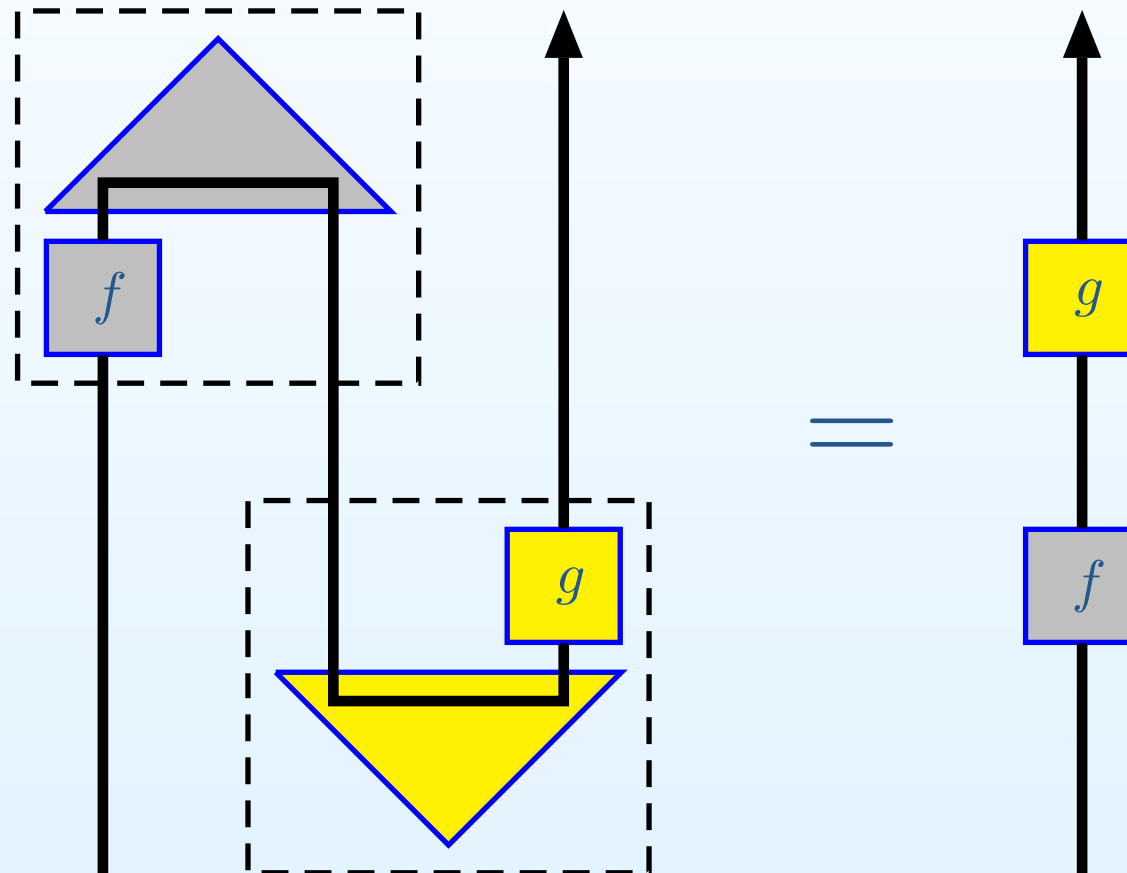
● Copy-Cat Strategies and Information Flow

● Compositionality ctd



# Compositionality

The key algebraic fact from which teleportation (and many other protocols) can be derived.



Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow
- Projectors
- Decomposed

• **Compositionality**

Copy-Cat Strategies and Information Flow

• Compositionality ctd

# Compositionality ctd

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

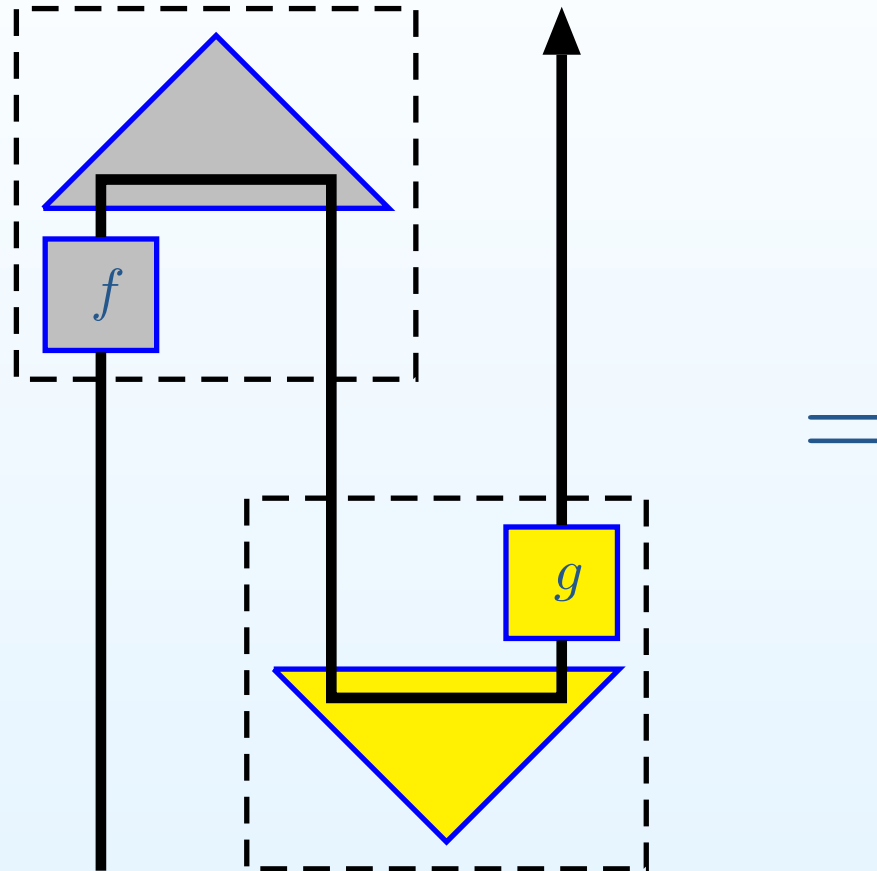
Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow
- Projectors
- Decomposed
- Compositionality

● Copy-Cut Strategies and Information Flow

● Compositionality ctd



# Compositionality ctd

[Introduction](#)

[Logic](#)

[From Proof Nets to  
Diagram Algebras](#)

[Temperley-Lieb Algebra](#)

[Characterizing Planarity](#)

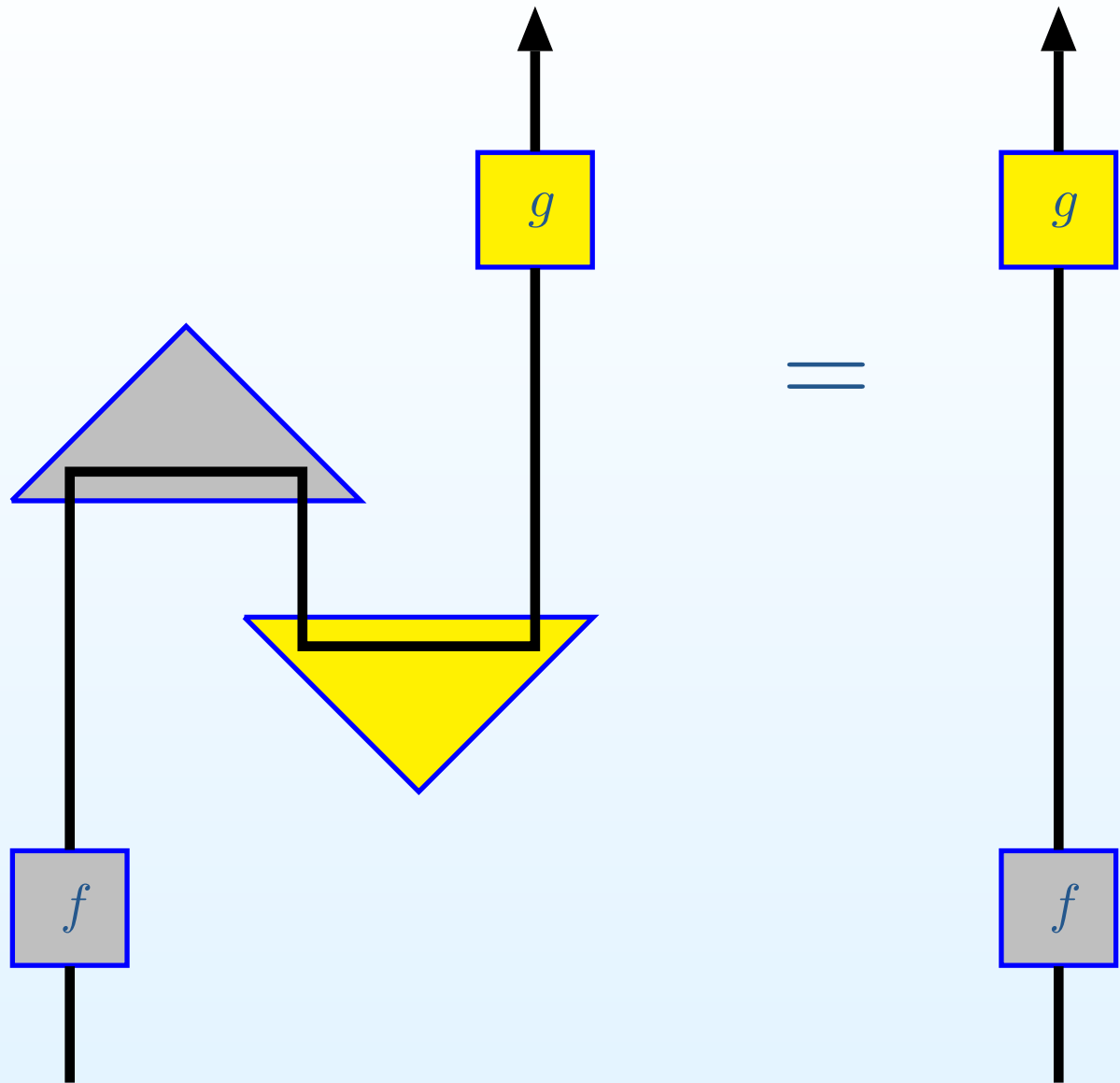
[Composition](#)

[Functional  
Computation: The  
Planar  \$\lambda\$ -calculus](#)

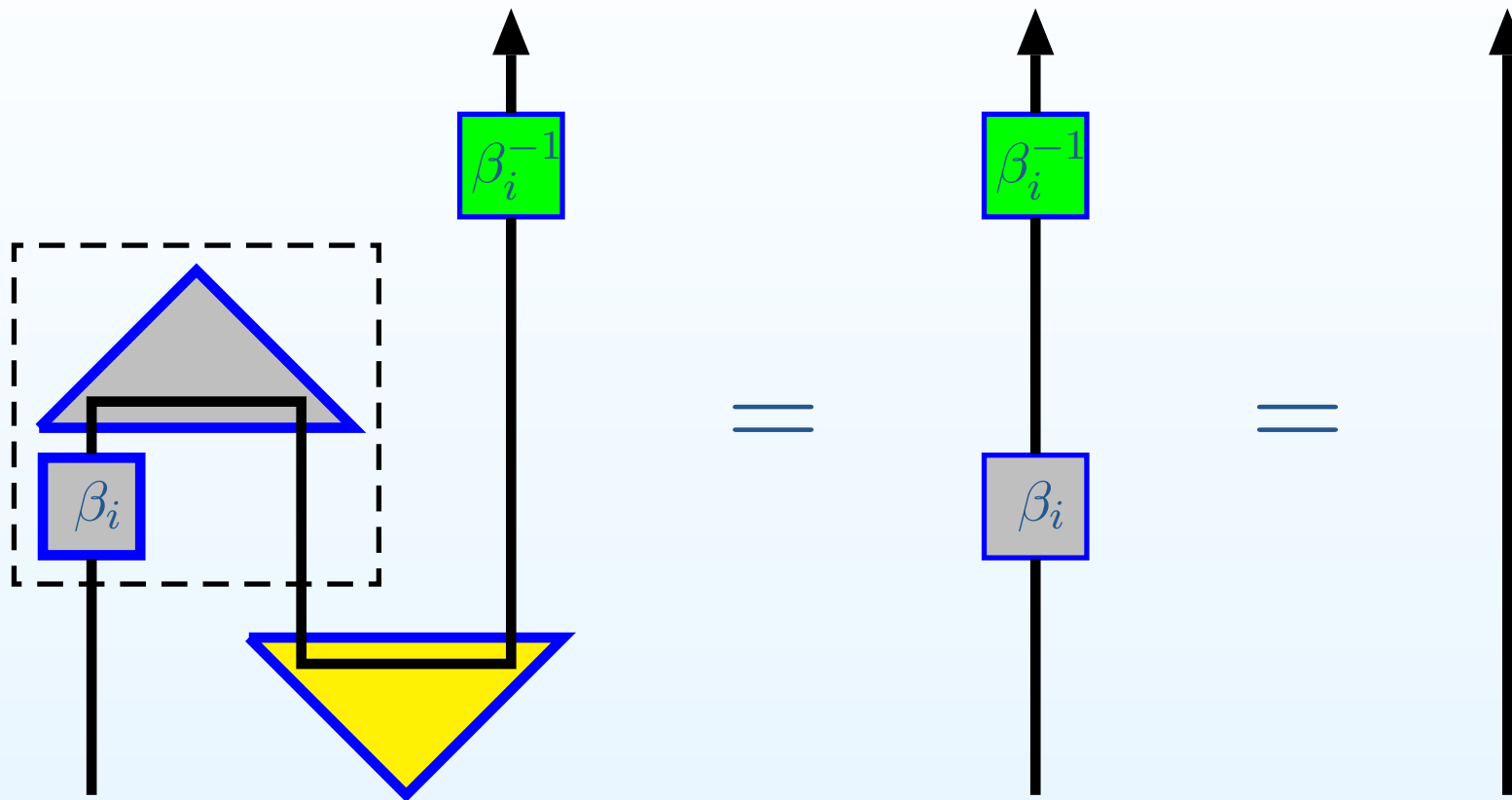
[Functional  
Computation:  
Non-Planar  
Combinators](#)

[Logic of Quantum  
Information Flow](#)

- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow
- Projectors
- Decomposed
- Compositionality



# Teleportation diagrammatically



# It's Logic!

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

- Quantum Entanglement
- From 'paradox' to 'feature': Teleportation
- What is the output?
- Follow the line!
- Graphical Calculus for Information Flow
- Projectors Decomposed
- Compositionality

The graphical calculus can be seen as a **calculus of proofs** for a certain **logic** — which is highly non-classical, (in particular **resource-sensitive**, so e.g. it builds in 'No Cloning'), but also very different from the Birkhoff-von Neumann quantum logic.

Simplification of diagrams — 'straightening out the lines' — corresponds to **normalization** or **cut-elimination** of proofs.

Introduction

Logic

From Proof Nets to  
Diagram Algebras

Temperley-Lieb Algebra

Characterizing Planarity

Composition

Functional  
Computation: The  
Planar  $\lambda$ -calculus

Functional  
Computation:  
Non-Planar  
Combinators

Logic of Quantum  
Information Flow

**Cloning**

- Cloning vs. Copy-cat
- Some References

# Cloning

## Cloning vs. Copy-cat

Copy-cat is **linear copying**: swapping  $A \leftrightarrow A$ , rather than **cloning**  
 $A \rightarrow A, A$ .

## Cloning vs. Copy-cat

Copy-cat is **linear copying**: swapping  $A \leftrightarrow A$ , rather than **cloning**  
 $A \rightarrow A, A$ .

- In **Logic**, Cloning corresponds to the Contraction rule

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

and takes us from Linear to Classical Logic.

## Cloning vs. Copy-cat

Copy-cat is **linear copying**: swapping  $A \leftrightarrow A$ , rather than **cloning**  
 $A \rightarrow A, A$ .

- In **Logic**, Cloning corresponds to the Contraction rule

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

and takes us from Linear to Classical Logic.

- In **Computation**, Cloning allows us to define combinators such as

$$\mathbf{W}xy = xyy$$

and takes us from linear-time to universal computational power.

## Cloning vs. Copy-cat

Copy-cat is **linear copying**: swapping  $A \leftrightarrow A$ , rather than **cloning**  
 $A \rightarrow A, A$ .

- In **Logic**, Cloning corresponds to the Contraction rule

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}$$

and takes us from Linear to Classical Logic.

- In **Computation**, Cloning allows us to define combinators such as

$$\mathbf{W}xy = xyy$$

and takes us from linear-time to universal computational power.

- In Physics, Cloning can be used to express the passage from quantum to classical: **given the choice of a basis**, we can define a linear map

$$\mathcal{H} \longrightarrow \mathcal{H} \otimes \mathcal{H}.$$

## Some References

Papers available from my webpages

<http://web.comlab.ox.ac.uk/oucl/work/samson.abramsky/>

- Abramsky, S. and Coecke, B. (2004) *A categorical semantics of quantum protocols*. Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LiCS'04), IEEE Computer Science Press.
- Abramsky, S. (2005) *Abstract Scalars, Loops, and Free Traced and Strongly Compact Closed Categories*. In Proceedings of CALCO 2005, Springer LNCS Vol. 3629, 1–31, 2005.
- S. Abramsky, Temperley-Lieb algebra: From knot theory to logic and computation via quantum mechanics. To appear in: *Mathematics of Quantum Computing and Technology*, ed. Chen, Kauffman and Lomonaco. Taylor and Francis, 2007.
- S. Abramsky, Information, Processes and Games. To appear in: *Handbook of the Philosophy of Information*, ed. P. Adriaans and J. van Benthem, Elsevier.