Reconsidering MacLane (again): algorithms for coherence ...

Peter M. Hines

York – Maths Dept. – Nov. 2013

Part (II) of a trilogy

This is a sequel to the talk of 16/10/2013.

What will be assumed:

- The definition of a category.
- The definitions of diagrams & functors.
- A rough idea about what tensors are.
- A very vague recollection of what I talked about last time.

The story so far ...

MacLane's theorem is possibly the most relied-upon theorem in category theory.

There is a 'mismatch' between:

1. The formal statement.
2. The informal statement.

Some areas are aware of this ... others less so.

This confusion seems to be due to MacLane himself.

MacLane's theorem – the general area

The topic is **tensors** on categories:

$$\_ \otimes \_ : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$$

The *informal* version is used to simplify associativity:

Associativity *up to isomorphism*

$$\begin{tikzcd} A \otimes (B \otimes C) \arrow[rr, bend left, "\tau_{A,B,C}"] & & (A \otimes B) \otimes C \arrow[ll, bend left, "\tau^{-1}_{A,B,C}"] \end{tikzcd}$$

is treated as a *strict equality*

$$\begin{tikzcd} A \otimes B \otimes C \arrow[rr, dash, "="] & & A \otimes B \otimes C \end{tikzcd}$$

Formal vs. Informal

- **(Correct ...)** Every diagram *in the image of* ***MacLane's substitution functor*** commutes.
- **(Incorrect ...)** Every *canonical* diagram commutes.

Canonical diagrams have arrows built using:

- Associativity isomorphisms, $\tau : X \otimes (Y \otimes Z) \to (X \otimes Y) \otimes Z$
- Identity arrows $1_X : X \to X$
- Tensors $\_ \otimes \_$
- Inverses $(\ )^{-1}$

Where the problem arises:

The **informal statement** true iff

MacLane's **substitution functor**

$$\mathcal{W}Sub : (W, \Box) \to (\mathcal{C}, \otimes)$$

is an embedding – an **epic functor**.

This is not always the case!

A simple example ...

The symmetric group on $\mathbb{N}$ is a **single-object category**.

A tensor derived from the Cantor pairing:

- **The tensor:**

$$(f \star g)(n) = \begin{cases} 2.f\left(\frac{n}{2}\right) & n \text{ even}, \\ 2.g\left(\frac{n-1}{2}\right)+1 & n \text{ odd}. \end{cases}$$

- **The associativity isomorphism:**

$$\tau(n) = \begin{cases} 2n & n\ (mod\ 2) = 0, \\ n+1 & n\ (mod\ 4) = 1, \\ \frac{n-1}{2} & n\ (mod\ 4) = 3. \end{cases}$$

I remember it well, in the Hilbert hotel

A large class of counterexamples

Tensors $(\_ \star \_)$ on the natural numbers $\mathbb{N}$

(– treated as a single-object category).

are equivalent to **self-similar structures**

i.e. isomorphisms
$$\begin{tikzcd} \mathbb{N} \arrow[r, bend left, "\rhd"] & \mathbb{N} \uplus \mathbb{N} \arrow[l, bend left, "\lhd"] \end{tikzcd}$$

derived from ‘Hilbert Hotel’ style reasoning.

Fixing a hole, where the strain comes in

What can be done about this?

1. Build 'equivalent' categories where all canonical diagrams commute.
2. Provide a coherence theorem & strictification procedure for self-similarity.
3. Give a decision procedure for commutativity of canonical diagrams.

These three solutions are very closely related[1].

[1] *after a little bit of work ...*

A reminder: MacLane's theorem

Assume a monoidal category $(\mathcal{C}, \otimes)$, with *generating object* S

MacLane's theorem relies on a functor

$$\mathcal{W}Sub : (\mathcal{W}, \Box) \to (C, \otimes)$$

MacLane's theorem (formal version)

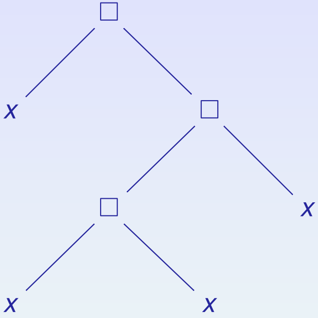
Every diagram in $\mathcal{C}$

that is the image of a diagram in $\mathcal{W}$

may be guaranteed to commute.

A reminder - the source of the functor

The category $\mathcal{W}$ is based on (non-empty) *binary trees*.



- **Leaves** labelled by x,
- **Branchings** labelled by $\square$.

The **rank** of a tree is the number of leaves.

A posetal category of trees

MacLane's category $\mathcal{W}$.

- **(Objects)** All non-empty binary trees.
- **(Arrows)** A **unique** arrow between any two trees *of the same rank*.

  — write this as $(v \leftarrow u) \in \mathcal{W}(u, v)$.

Key points:

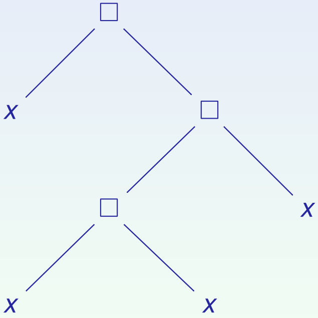
1. $(\_\square\_)$ is a tensor on $\mathcal{W}$.
2. $\mathcal{W}$ is **posetal** — all diagrams over $\mathcal{W}$ commute.

MacLane's substitution functor

On objects:

- $\mathcal{W}Sub(x) = S$,
- $\mathcal{W}Sub(u\Box v) = \mathcal{W}Sub(u) \otimes \mathcal{W}Sub(v)$.

An object of $\mathcal{W}$:

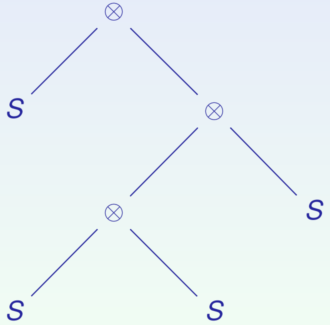


An inductively defined functor (I)

On objects:

- $\mathcal{W}Sub(x) = S$,
- $\mathcal{W}Sub(u \Box v) = \mathcal{W}Sub(u) \otimes \mathcal{W}Sub(v)$.

An object of $\mathcal{C}$:



An inductively defined functor (II)

On arrows:

- $\mathcal{W}Sub(u \leftarrow u) = 1_{\_}$.
- $\mathcal{W}Sub(a\Box v \leftarrow a\Box u) = 1_{\_} \otimes \mathcal{W}Sub(v \leftarrow u)$.
- $\mathcal{W}Sub(v\Box b \leftarrow u\Box b) = \mathcal{W}Sub(v \leftarrow u) \otimes 1_{\_}$.
- $\mathcal{W}Sub((a\Box b)\Box c \leftarrow a\Box(b\Box c)) = \tau_{\_,\_,\_}$.

The coherence condition ...

MacLane's Pentagon condition ensures $\mathcal{W}Sub$ is a functor.

An inductively defined functor (II)

On arrows:

- $\mathcal{W}Sub(u \leftarrow u) = 1_{\_}$.
- $\mathcal{W}Sub(a\Box v \leftarrow a\Box u) = 1_{\_} \otimes \mathcal{W}Sub(v \leftarrow u)$.
- $\mathcal{W}Sub(v\Box b \leftarrow u\Box b) = \mathcal{W}Sub(v \leftarrow u) \otimes 1_{\_}$.
- $\mathcal{W}Sub((a\Box b)\Box c \leftarrow a\Box(b\Box c)) = \tau_{\_,\_,\_}$.

The coherence condition ...

MacLane's Pentagon condition ensures $\mathcal{W}Sub$ is a functor.

The root of the problem:

We have a functor $\mathcal{W}Sub : (\mathcal{W}, \Box) \to (\mathcal{C}, \otimes)$.

- Every **object** of $\mathcal{C}$ is the image of an object of $\mathcal{W}$
- Every **canonical arrow** of $\mathcal{C}$ is the image of an arrow of $\mathcal{W}$
- Every **diagram** over $\mathcal{W}$ commutes.
- The image of **every diagram** in $(W, \Box)$ **commutes**
- Every canonical diagram is of this form
  precisely when $\mathcal{W}Sub$ is an embedding.

The root of the problem:

We have a functor $\mathcal{W}Sub : (\mathcal{W}, \Box) \rightarrow (\mathcal{C}, \otimes)$.

- Every **object** of $\mathcal{C}$ is the image of an object of $\mathcal{W}$
- Every **canonical arrow** of $\mathcal{C}$ is the image of an arrow of $\mathcal{W}$
- Every **diagram** over $\mathcal{W}$ commutes.
- The image of **every diagram** in $(W, \Box)$ **commutes**
- Every canonical diagram is of this form
  precisely when $\mathcal{W}Sub$ is an embedding.

The root of the problem:

We have a functor $\mathcal{W}Sub : (\mathcal{W}, \Box) \to (\mathcal{C}, \otimes)$.

- Every **object** of $\mathcal{C}$ is the image of an object of $\mathcal{W}$
- Every **canonical arrow** of $\mathcal{C}$ is the image of an arrow of $\mathcal{W}$
- Every **diagram** over $\mathcal{W}$ commutes.
- The image of **every diagram** in $(W, \Box)$ **commutes**
- Every canonical diagram is of this form
precisely when $\mathcal{W}Sub$ is an embedding.

Approach I

Building categories where

all canonical diagrams commute.

Given a **badly-behaved** category,

we will build a ***well-behaved*** version.

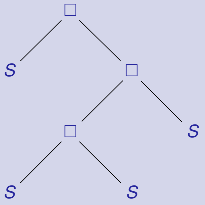
Given a (monogenic) monoidal category $(\mathcal{C}, \otimes)$:

We will construct a ‘closely related’ category

for which MacLane’s functor is an *embedding*.

Constructing $Plat_{\mathcal{C}}$

Objects are free binary trees



Leaves labelled by $S \in Ob(\mathcal{C})$,

Branchings labelled by $\Box$.

There is an **instantiation map** $Inst : Ob(Plat_{\mathcal{C}}) \rightarrow Ob(\mathcal{C})$

$$S\Box((S\Box S)\Box S) \;\mapsto\; S \otimes ((S \otimes S) \otimes S)$$

This is not just a matter of syntax!

What about arrows?

Homsets are copies of homsets of $\mathcal{C}$

Given trees T_1, T_2,

$$Plat_{\mathcal{C}}(T_1, T_2) = \mathcal{C}(Inst(T_1), Inst(T_2))$$

Composition is inherited from $\mathcal{C}$ in the obvious way.

The tensor $(\_\square\_) : Plat_{\mathcal{C}} \times Plat_{\mathcal{C}} \rightarrow Plat_{\mathcal{C}}$

$$\left.\begin{array}{l} A \xrightarrow{f} X \\ \\ \\ B \xrightarrow{g} Y \end{array}\right\} \quad A\square X \xrightarrow{f\square g} B\square Y$$

The tensor of $Plat_{\mathcal{C}}$ is

- **(Objects)** A free formal pairing, $A\square B$,
- **(Arrows)** Inherited from $(\mathcal{C}, \otimes)$, so $f\square g \overset{def.}{=} f \otimes g$.

Some properties of the platonic ideal ...

1. The functor

$$\mathcal{W}Sub : (\mathcal{W}, \Box) \to (Plat_{\mathcal{C}}, \Box)$$

is always **monic.**

2. As a corollary:

All canonical diagrams of $(Plat_{\mathcal{C}}, \Box)$ commute.

3. Instantiation defines an **epic** monoidal functor

$$Inst : (Plat_{\mathcal{C}}, \Box) \to (\mathcal{C}, \otimes)$$

through which McL'.s substitution functor always factors.

Some properties of the platonic ideal ...

1. The functor

$$\mathcal{W}Sub : (\mathcal{W}, \Box) \to (Plat_{\mathcal{C}}, \Box)$$

   is always **monic.**

2. As a corollary:

   All canonical diagrams of $(Plat_{\mathcal{C}}, \Box)$ commute.

3. Instantiation defines an **epic** monoidal functor

$$Inst : (Plat_{\mathcal{C}}, \Box) \to (\mathcal{C}, \otimes)$$

   through which McL'.s substitution functor always factors.

Some properties of the platonic ideal ...

1. The functor

$$\mathcal{W}Sub : (\mathcal{W}, \Box) \to (Plat_{\mathcal{C}}, \Box)$$

is always **monic.**

2. As a corollary:

All canonical diagrams of $(Plat_{\mathcal{C}}, \Box)$ commute.

3. Instantiation defines an **epic** monoidal functor

$$Inst : (Plat_{\mathcal{C}}, \Box) \to (\mathcal{C}, \otimes)$$

through which McL'.s substitution functor always factors.

A monic / epic decomposition

MacLane's substitution functor always factors through the platonic ideal:

$$\begin{tikzcd}
(\mathcal{W}, \Box) \arrow[r, "\overset{\text{(monic)}}{\mathcal{W}Sub_{_}}"] \arrow[dr, "\mathcal{W}Sub_{_}"'] & (Plat_{\mathcal{C}}, \Box) \arrow[d, "Inst\ \text{(epic)}"] \\
& (\mathcal{C}, \otimes)
\end{tikzcd}$$

This gives a monic / epic decomposition of his functor.

Approach II

Give a ‘strictification procedure’

for self-similarity $S \cong S \otimes S$.

To be compared & contrasted with

MacLane’s ‘strictification procedure’ for associativity.

What is **strictification**?

Given a structural property of a category:

Associativity	$A \otimes (B \otimes C)$	$\cong$	$(A \otimes B) \otimes C$
Symmetry	$A \otimes B$	$\cong$	$B \otimes A$
Distributivity	$A \otimes (B \oplus C)$	$\cong$	$(A \otimes B) \oplus (A \otimes C)$
Self-similarity	S	$\cong$	$S \otimes S$
Interchange	$(A \otimes B) \star (C \otimes D)$	$\cong$	$(A \star C) \otimes (B \star D)$

We (attempt to) form a **strict** version of the same category.

What is **strictification**?

Strictification gives an “equivalent” category

Associativity	$A \otimes (B \otimes C)$	$=$	$(A \otimes B) \otimes C$
Symmetry	$A \otimes B$	$=$	$B \otimes A$
Distributivity	$A \otimes (B \oplus C)$	$=$	$(A \otimes B) \oplus (A \otimes C)$
Self-similarity	S	$=$	$S \otimes S$
Interchange	$(A \otimes B) \star (C \otimes D)$	$=$	$(A \star C) \otimes (B \star D)$

where **isomorphisms** are replaced by ~~equalities~~ **identities.**

Let me tell you what I want

What would we **like** from strictification?

1. All *canonical isomorphisms* to be replaced by *identities*.
2. This process should be *functorial*.
3. There should be no ‘side-effects’.

The commutativity of a diagram in the *strict category*

$$\Longleftrightarrow$$

The commutativity of ‘equivalent’ diagram(s)
in the *non-strict version*.

Let me tell you what I want

What would we **like** from strictification?

1. All *canonical isomorphisms* to be replaced by *identities*.
2. This process should be *functorial*.
3. There should be no 'side-effects'.

The commutativity of a diagram in the *strict category*

$$\Longleftrightarrow$$

The commutativity of 'equivalent' diagram(s)
in the *non-strict version*.

Let me tell you what I want

What would we **like** from strictification?

1. All *canonical isomorphisms* to be replaced by *identities*.
2. This process should be *functorial*.
3. There should be no ‘side-effects’.

The commutativity of a diagram in the *strict category*

$$\Longleftrightarrow$$

The commutativity of ‘equivalent’ diagram(s) in the *non-strict version*.

You can't always get what you want

The definition of *equivalent* is very subtle!

Strictification often has 'side effects'

Strictifying **Distributivity**

$$A \otimes (B \oplus C) = (A \otimes B) \oplus (A \otimes C)$$

forces strict **symmetry** for $(\_ \oplus \_)$.

You can't always get what you want

The definition of *equivalent* is very subtle!

Strictification often has 'side effects'

Strictifying **symmetry**

$$A \otimes B = B \otimes A$$

brings on **many changes.**

You can't always get what you want

The definition of *equivalent* is very subtle!

Strictification often has 'side effects'

Strictifying **associativity**

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C$$

maps **single-object** categories to **multi-object** categories.

You can't always get what you want

The definition of *equivalent* is very subtle!

Strictification often has 'side effects'

Strictifying **self-similarity**

$$S = S \otimes S$$

forces associativity **up to isomorphism.**

Not everything can be strict ...

Not all these procedures are compatible.

The 'No Simultaneous Strictness' Theorem

One cannot have both

(I) Associativity $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$

(II) Self-Similarity $S \cong S \otimes S$

as strict equalities.

There are no strict tensors on non-trivial monoids!

How to strictify self-similarity

A simple, almost painless, procedure (I)

- Start with a monogenic category $(\mathcal{C}, \otimes)$, generated by a self-similar object

$$\begin{tikzcd} S \arrow[r, bend left, "\rhd"] & S \otimes S \arrow[l, bend left, "\lhd"] \end{tikzcd}$$

- Construct its platonic ideal $(Plat_{\mathcal{C}}, \square)$
- For *every* object A, define a pair of isomorphisms:

$$\begin{tikzcd} S \arrow[r, bend left, "\rhd_A"] & A \arrow[l, bend left, "\lhd_A"] \end{tikzcd}$$

The **generalised code** / **decode** arrows.

A simple, almost painless, procedure (I)

- Start with a monogenic category $(\mathcal{C}, \otimes)$, generated by a self-similar object

$$\begin{tikzcd} S \arrow[r, bend left, "\rhd"] & S \otimes S \arrow[l, bend left, "\lhd"] \end{tikzcd}$$

- Construct its platonic ideal $(Plat_{\mathcal{C}}, \Box)$

- For *every* object A, define a pair of isomorphisms:

$$\begin{tikzcd} S \arrow[r, bend left, "\rhd_A"] & A \arrow[l, bend left, "\lhd_A"] \end{tikzcd}$$

The **generalised code** / **decode** arrows.

A simple, almost painless, procedure (I)

- Start with a monogenic category $(\mathcal{C}, \otimes)$, generated by a self-similar object

$$\begin{tikzcd} S \arrow[rr, bend left, "\rhd"] & & S \otimes S \arrow[ll, bend left, "\lhd"] \end{tikzcd}$$

- Construct its platonic ideal $(Plat_{\mathcal{C}}, \Box)$
- For *every* object A, define a pair of isomorphisms:

$$\begin{tikzcd} S \arrow[rr, bend left, "\rhd_A"] & & A \arrow[ll, bend left, "\lhd_A"] \end{tikzcd}$$

The **generalised code** / **decode** arrows.

Generalised code / decode arrows

An inductive definition:

- For the generating object,

$$\lhd_S = 1_S = \rhd_S$$

- For arbitrary objects A, B, we

  define $\lhd_{A\Box B}$ in terms of $\lhd_A$ and $\lhd_B$.

$$\begin{tikzcd}
A \otimes B \arrow[d, "\lhd_A \Box \lhd_B"'] \arrow[dr, dotted, "\lhd_{A\Box B}"] & \\
S \Box S \arrow[r, "\lhd"'] & S
\end{tikzcd}$$

A simple,almost painless, procedure (II)

- This gives, for all objects A,
  a unique pair of inverse arrows

$$
\begin{tikzcd}
S \arrow[r, bend left, "\rhd_A"] & A \arrow[l, bend left, "\lhd_A"]
\end{tikzcd}
$$

- Use these to define an **endofunctor** $\Phi : Plat_{\mathcal{C}} \to Plat_{\mathcal{C}}$.

A simple,almost painless, procedure (II)

- This gives, for all objects A,
  a unique pair of inverse arrows

$$\begin{tikzcd} S \arrow[r, bend left, "\rhd_A"] & A \arrow[l, bend left, "\lhd_A"] \end{tikzcd}$$

- Use these to define an **endofunctor** $\Phi : Plat_{\mathcal{C}} \rightarrow Plat_{\mathcal{C}}$.

The **type-erasing** endofunctor

- **Objects**

$$\Phi(A) = S \quad , \quad \text{for all objects } A$$

- **Arrows**

$$\begin{tikzcd} A \arrow[r, "f"] & B \arrow[d, "\lhd_B"] \\ S \arrow[u, "\rhd_A"] \arrow[r, "\Phi(f)"'] & S \end{tikzcd}$$

- **Functoriality** is trivial ...

As a final step:

Define a tensor $(\_ \star \_)$ on $\mathcal{C}(S, S)$ by

$$\begin{tikzcd} S \otimes S \arrow[r, "t \otimes u"] & S \otimes S \arrow[d, "\lhd"] \\ S \arrow[u, "\rhd"] \arrow[r, "t \star u"'] & S \end{tikzcd}$$

$(C(S, S), \_ \star \_)$ is a single-object monoidal category!

Type-erasing as a monoidal functor

- Recall, $Plat_{\mathcal{C}}(S, S) \cong \mathcal{C}(S, S)$.
- Up to this obvious isomorphism,

$$\Phi : (Plat_{\mathcal{C}}, \Box) \to (\mathcal{C}(S, S), \star)$$

is a monoidal functor.

What we have ...

A monoidal functor from $Plat_{\mathcal{C}}$

to a strictly self-similar monoidal category.

— every canonical (for self-similarity) arrow is mapped to 1_S.

Type-erasing as a monoidal functor

- Recall, $Plat_{\mathcal{C}}(S,S) \cong \mathcal{C}(S,S)$.
- Up to this obvious isomorphism,

$$\Phi : (Plat_{\mathcal{C}}, \Box) \to (\mathcal{C}(S,S), \star)$$

is a monoidal functor.

What we have ...

A monoidal functor from $Plat_{\mathcal{C}}$

to a <u>strictly self-similar</u> monoidal category.

— every canonical (for self-similarity) arrow is mapped to 1_S.

A useful property

Basic Category Theory

diagram $\mathfrak{D}$ commutes $\Rightarrow$ diagram $\Phi(\mathfrak{D})$ commutes.

$\mathfrak{D}$

$$
\begin{tikzcd}
 & v \arrow[dr, "g"] & \\
u \arrow[ur, "f"] \arrow[rr, "h"'] & & w
\end{tikzcd}
$$

$\Phi(\mathfrak{D})$

$$
\begin{tikzcd}
 & S \arrow[dr, "\Phi(g)"] & \\
S \arrow[ur, "\Phi(f)"] \arrow[rr, "\Phi(h)"'] & & S
\end{tikzcd}
$$

As above, so below

In this case ...

diagram $\mathfrak{D}$ commutes $\Leftrightarrow$ diagram $\Phi(\mathfrak{D})$ commutes.

$\mathfrak{D}$

$$\begin{tikzcd}
 & v \arrow[dr, "g"] \arrow[dd, shift left, "\lhd_v"] & \\
u \arrow[ur, "f"] \arrow[rr, "h"'] \arrow[dd, shift left, "\lhd_u"] & & w \arrow[dd, shift left, "\lhd_w"] \\
 & S \arrow[uu, shift left, "\rhd_v"] \arrow[dr, "\Phi(g)"] & \\
S \arrow[uu, shift left, "\rhd_u"] \arrow[ur, "\Phi(f)"] \arrow[rr, "\Phi(h)"'] & & S \arrow[uu, shift left, "\rhd_w"]
\end{tikzcd}$$

$\Phi(\mathfrak{D})$

An application:

Simplifying proofs in published papers.

(P.H. 2013) *Types and forgetfulness in categorical linguistics and quantum mechanics*, in Sadrzadeh, Heunen, Greffenstette (ed.s), **Categorical Information Flow in Physics and Linguistics**, O.U.P.

The theorem:

Any self-similar structure $(S, \lhd, \rhd)$ in a symmetric monoidal category defines a (unitless) Frobenius algebra.

The interpretation: Semantic models of conjunction, in computational linguistics, satisfy the same formal axioms as categorical models of measurement in quantum mechanics.

An application (cont.)

The key step:

Proving this diagram commutes:

$$\begin{tikzcd}
 & S \otimes S \arrow[dl, "\lhd"'] \arrow[dr, "\rhd \otimes 1_S"] & \\
S \arrow[d, "\tau^{-1}"'] & & (S \otimes S) \otimes S \arrow[d, "\tau^{-1}_{S,S,S}"] \\
S \arrow[dr, "\rhd"'] & & S \otimes (S \otimes S) \arrow[dl, "1_S \otimes \lhd"] \\
 & S \otimes S &
\end{tikzcd}$$

The key step:

Applying Φ gives:

$$
\begin{tikzcd}
 & S \arrow[dl, "1_S"'] \arrow[dr, "1_S \star 1_S"] & \\
S \arrow[d, "\tau^{-1}"'] & & S \arrow[d, "\tau^{-1}"] \\
S \arrow[dr, "1_S"'] & & S \arrow[dl, "1_S \star 1_S"] \\
 & S &
\end{tikzcd}
$$

An application (cont.)

The key step:

Simplifying slightly:

$$\begin{tikzcd}
& S \arrow[dl, "1_S"'] \arrow[dr, "1_S"] & \\
S \arrow[d, "\tau^{-1}"'] & & S \arrow[d, "\tau^{-1}"] \\
S \arrow[dr, "1_S"'] & & S \arrow[dl, "1_S"] \\
& S &
\end{tikzcd}$$

The key step:

One more time:

$$\begin{tikzcd} S \arrow[d, bend right, "\tau^{-1}"'] \arrow[d, bend left, "\tau^{-1}"] \\ S \end{tikzcd}$$

This commutes(!), hence the original diagram also commutes.

This is simpler than the published proof.

Approach (III)

A decision procedure

for commutativity of

canonical diagrams

Deciding whether a canonical diagram commutes

("They all do" is not a valid answer!)

We do this for single-object categories

– the general case follows –

The Platonic ideal of an untyped category

the platonic ideal of a single-object category $\mathcal{C}$

- is **monogenic.**
- has **infinitely many objects.**
- has a **self-similar** generating object $S \cong S \otimes S$.

We have defined two functors

$$\begin{tikzcd} Plat_{\mathcal{C}} \arrow[r, bend left, "\Phi"] \arrow[r, bend right, "Inst"'] & \mathcal{C} \end{tikzcd}$$

In this case, these are equal.

When do untyped diagrams commute?

For any canonical diagram $\mathfrak{U}$ over $(\mathcal{C}, \star)$

- All nodes are the single object S.
- All arrows are built from $(\_ \star \_)$, 1_S, τ, $(\ )^{-1}$.

The key fact:

The diagram $\mathfrak{U}$ commutes precisely when

it is the image under Φ of some diagram in $Plat_{\mathcal{C}}$.

Question: Can we decide when such a diagram exists?

The key fact!

Recall that the functors

$$\begin{tikzcd} \mathit{Plat}_{\mathcal{C}} \arrow[r, bend left, "\Phi"] \arrow[r, bend right, "\mathit{Inst}"'] & \mathcal{C} \end{tikzcd}$$

are <u>equal</u> in this setting.

It is *much* easier to ask:

"Is diagram $\mathfrak{U}$ of the form $\mathit{Inst}(\mathfrak{T})$,

for some $\mathfrak{T}$ over $\mathit{Plat}_{\mathcal{C}}$? "

From 'untyped' to 'typed'

Key question: Is $\mathfrak{U}$ type-able?

Can we consistently replace:

Diagram $\mathfrak{U}$	*Diagram* $\mathfrak{T}$
Every object S	*by binary tree of variable symbols*
Every identity 1_S	*by some identity on such trees.*
Each untyped tensor $(\_ \star \_)$	*by the typed tensor* $(\_\Box\_)$
Each untyped assoc. iso. τ	*by some typed assoc. iso.* $\tau_{X,Y,Z}$

to give a new well-formed diagram $\mathfrak{T}$?

A useful fact:

In the category $\mathit{Plat}_{\mathcal{C}}$, there is **at most one** canonical arrow, between any two objects.

In a connected commuting diagram

The ‘typing’ at a single object determines the ‘typing’ of the whole diagram.

An example: the untyped pentagon

$$\begin{tikzcd}
 & S \arrow[dr, "1\star\tau"] \arrow[ddl, "\tau"'] & \\
 & & S \arrow[dd, "\tau"] \\
S \arrow[ddr, "\tau"'] & & \\
 & & S \arrow[dl, "\tau\star 1"] \\
 & S &
\end{tikzcd}$$

An example: the untyped pentagon

$$
\begin{tikzcd}
 & S \arrow[dl, "\tau"'] \arrow[dr, "1\star\tau"] & \\
S \arrow[ddr, "\tau"'] & & S \arrow[d, "\tau"] \\
 & & S \arrow[dl, "\tau\star 1"] \\
 & S &
\end{tikzcd}
$$

Typing the untyped:

$$
\begin{tikzcd}
{} & A\Box(B\Box(C\Box D)) \arrow[rd, "1_A\Box\tau_{B,C,D}"] \arrow[ldd, dotted, "\tau"'] & {} \\
{} & {} & A\Box((B\Box C)\Box D) \arrow[d, dotted, "\tau"] \\
S \arrow[rd, dotted, "\tau"'] & {} & S \arrow[ld, dotted, "\tau"] \\
{} & S & {}
\end{tikzcd}
$$

Where A, B, C, D are variable symbols over binary trees.

Typing the untyped:

$$\begin{tikzcd}
 & A\Box(B\Box(C\Box D)) \arrow[r, "1_A\Box\tau_{B,C,D}"] \arrow[dl, dotted, "\tau"'] & A\Box((B\Box C)\Box D) \arrow[d, dotted, "\tau_{A,B\Box C,D}"] \\
S \arrow[dr, dotted, "\tau"'] & & (A\Box(B\Box C))\Box D \arrow[dl, dotted, "\tau\star 1"] \\
 & S &
\end{tikzcd}$$

Where A, B, C, D are variable symbols over binary trees.

Typing the untyped:

$$
\begin{tikzcd}
 & A\square(B\square(C\square D)) \arrow[dl, dotted, "\tau"'] \arrow[r, "1_A\square\tau_{B,C,D}"] & A\square((B\square C)\square D) \arrow[d, "\tau_{A,B\square C,D}"] \\
S \arrow[dr, dotted, "\tau"'] & & (A\square(B\square C))\square D \arrow[dl, "\tau_{A,B,C}\star 1_D"] \\
 & ((A\square B)\square C)\square D &
\end{tikzcd}
$$

Where A, B, C, D are variable symbols over binary trees.

Typing the untyped:

$$
\begin{tikzcd}
 & A\Box(B\Box(C\Box D)) \arrow[dr, "1_A\Box\tau_{B,C,D}"] \arrow[ddl, "\tau_{A,B,C\Box D}"'] & \\
 & & A\Box((B\Box C)\Box D) \arrow[dd, "\tau_{A,B\Box C,D}"] \\
(A\Box B)\Box(C\Box D) \arrow[ddr, dotted, "\tau"'] & & \\
 & & (A\Box(B\Box C))\Box D \arrow[dl, "\tau_{A,B,C}\Box 1_D"] \\
 & ((A\Box B)\Box C)\Box D &
\end{tikzcd}
$$

Where A, B, C, D are variable symbols over binary trees.

Typing the untyped:

$$
\begin{tikzcd}
 & A\Box(B\Box(C\Box D)) \arrow[dl, "\tau_{A,B,C\Box D}"'] \arrow[dr, "1_A\Box\tau_{B,C,D}"] & \\
(A\Box B)\Box(C\Box D) \arrow[ddr, "\tau_{A\Box B,C,D}"'] & & A\Box((B\Box C)\Box D) \arrow[d, "\tau_{A,B\Box C,D}"] \\
 & & (A\Box(B\Box C))\Box D \arrow[dl, "\tau_{A,B,C}\Box 1_D"] \\
 & ((A\Box B)\Box C)\Box D &
\end{tikzcd}
$$

Where A, B, C, D are variable symbols over binary trees.

Not all diagrams are typeable

We <u>cannot</u> type:

$$\begin{tikzcd} S \arrow[r, bend left, "\tau"] \arrow[r, bend right, "1 \star \tau"'] & S \end{tikzcd}$$

Using variable symbols $X, Y, Z, \ A, B, C, D$:

$$\begin{tikzcd} X\Box(Y\Box Z) \arrow[r, "\tau_{X,Y,Z}"] \arrow[d, equal] & (X\Box Y)\Box Z \arrow[d, equal] \\ A\Box(B\Box(C\Box D)) \arrow[r, "1_A\Box\tau_{B,C,D}"'] & A\Box((B\Box C)\Box D) \end{tikzcd}$$

This is a fatal disagreement, in the sense
of Robinson's unification algorithm.

Not all diagrams are typeable

We cannot type:

$$\begin{tikzcd} S \arrow[r, bend left, "\tau"] \arrow[r, bend right, "1 \star \tau"'] & S \end{tikzcd}$$

Using variable symbols $X, Y, Z,\ A, B, C, D$:

$$\begin{tikzcd} X\Box(Y\Box Z) \arrow[r, "\tau_{X,Y,Z}"] \arrow[d, equal] & (X\Box Y)\Box Z \arrow[d, equal] \\ A\Box(B\Box(C\Box D)) \arrow[r, "1_A\Box\tau_{B,C,D}"'] & A\Box((B\Box C)\Box D) \end{tikzcd}$$

This is a fatal disagreement, in the sense of Robinson's unification algorithm.

The general case

Let $\mathfrak{U}$ be an arbitrary (canonical, untyped) diagram:

$$\begin{tikzcd}
 & S \arrow[rr, "d"] \arrow[dr, "c"] & & S \arrow[dl, "e"'] \arrow[dr, "f"] & \\
S \arrow[ur, "a"] \arrow[rr, "b"] \arrow[drr, "h"] & & S \arrow[rr, "g"] & & S \\
 & & S \arrow[urr, "k"] & &
\end{tikzcd}$$

The general case

Choose an arbitrary node:

$$\begin{tikzcd}
 & S \arrow[rr, "d"] \arrow[dr, "c"] & & S \arrow[dl, "e"'] \arrow[dr, "f"] & \\
\color{red}{S} \arrow[ur, "a"] \arrow[rr, "b"] \arrow[drr, "h"] & & S \arrow[rr, "g"] & & S \\
 & & S \arrow[urr, "k"] & &
\end{tikzcd}$$

Covering a diagram with loops

By replacing various isomorphisms by their inverses,

we may ‘cover’ $\mathfrak{U}$ with a finite set of distinct closed loops,

all starting / finishing at our distinguished node.

Covering a diagram with loops

Our diagram:

$$\begin{tikzcd}
 & S \arrow[rr, "d"] \arrow[dr, "c"] & & S \arrow[dl, "e"] \arrow[dr, "f"] & \\
S \arrow[ur, "a"] \arrow[rr, "b"] \arrow[drr, "h"] & & S \arrow[rr, "g"] & & S \\
 & & S \arrow[urr, "k"] & &
\end{tikzcd}$$

Loop L_1

$$\begin{tikzcd}
 & S \arrow[rr, dotted, "d"] \arrow[dr, "c"] & & S \arrow[dl, dotted, "e"'] \arrow[dr, dotted, "f"] & \\
{\color{red}S} \arrow[ur, "a"] \arrow[drr, dotted, "h"] & & S \arrow[ll, "b^{-1}"] \arrow[rr, dotted, "g"] & & S \\
 & & S \arrow[urr, dotted, "k"] & &
\end{tikzcd}$$

Covering a diagram with loops

Loop L_2

$$
\begin{tikzcd}
 & S \arrow[rr, "d"] \arrow[dr, dotted, "c"] & & S \arrow[dl, "e"] \arrow[dr, dotted, "f"] & \\
{\color{red} S} \arrow[ur, "a"] \arrow[rr, dotted, "b"] & & S \arrow[rr, "g"] & & S \arrow[dll, "k^{-1}"] \\
 & & S \arrow[ull, "h^{-1}"] & &
\end{tikzcd}
$$

Covering a diagram with loops

Loop L_3

$$\begin{tikzcd}
 & S \arrow[rr, "d"] \arrow[dr, dotted, "c"] & & S \arrow[dl, dotted, "e"'] \arrow[dr, "f"] & \\
S \arrow[ur, "a"] \arrow[drr, dotted, "h"] & & S \arrow[ll, "b^{-1}"] & & S \arrow[ll, "g^{-1}"] \\
 & & S \arrow[urr, dotted, "k"] & &
\end{tikzcd}$$

Together, these loops L_1, L_2, L_3 'cover' the diagram $\mathfrak{U}$.

Provided the diagram commutes, each of these closed loops is the identity.

Each closed loop

determines a binary tree of variable

symbols at the distinguished node.

Call these trees T_1, T_2, T_3 respectively.

Unifying typings

Typings T_1, T_2, T_3 are binary trees built up using:

- The operation $(\_\square\_)$,
- Variable symbols over objects of $Plat_{\mathcal{C}}$.

Taking care with variable names ...

We try to find T, the *most general unifier* of $\{T_1, T_2, T_3\}$

using **Robinson's Unification Algorithm.**

This exists **if and only if** the diagram commutes.

How complex is this algorithm?

- **Robinson (1965)**

  Exponentially complex $O(2^n)$ (in both time & space).

- **Paterson & Wegman (1978)**

  A linear $O(n)$ algorithm for unification.

- **Ružička & Prívara (1982)**

  Robinson's original algorithm is made 'almost linear'

  i.e. $O(n^{1+\epsilon})$ complexity, where $\epsilon = \frac{1}{Ack(n,n)}$.

Let's compare this with the alternative ...

How complex is this algorithm?

- **Robinson (1965)**

  Exponentially complex $O(2^n)$ (in both time & space).

- **Paterson & Wegman (1978)**

  A linear $O(n)$ algorithm for unification.

- **Ružička & Prívara (1982)**

  Robinson's original algorithm is made 'almost linear'

  i.e. $O(n^{1+\epsilon})$ complexity, where $\epsilon = \frac{1}{Ack(n,n)}$.

Let's compare this with the alternative ...

How complex is this algorithm?

- **Robinson (1965)**

  Exponentially complex $O(2^n)$ (in both time & space).

- **Paterson & Wegman (1978)**

  A linear $O(n)$ algorithm for unification.

- **Ružička & Prívara (1982)**

  Robinson's original algorithm is made 'almost linear'

  i.e. $O(n^{1+\epsilon})$ complexity, where $\epsilon = \frac{1}{Ack(n,n)}$.

Let's compare this with the alternative ...

How complex is this algorithm?

- **Robinson (1965)**

  Exponentially complex $O(2^n)$ (in both time & space).

- **Paterson & Wegman (1978)**

  A linear $O(n)$ algorithm for unification.

- **Ružička & Prívara (1982)**

  Robinson's original algorithm is made 'almost linear'

  i.e. $O(n^{1+\epsilon})$ complexity, where $\epsilon = \frac{1}{Ack(n,n)}$.

Let's compare this with the alternative ...

Back to playing with toys ...

Recall our 'toy example'

- Single object $\mathbb{N}$.
- Arrows: all bijections on $\mathbb{N}$.

The 'Cantor tensor'

We have a tensor $(\_ \star \_) : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$.

$$(f \star g)(n) = \begin{cases} 2.f\left(\frac{n}{2}\right) & n \text{ even}, \\ 2.g\left(\frac{n-1}{2}\right) + 1 & n \text{ odd}. \end{cases}$$

Associativity in the toy example

The associativity isomorphism is:

$$\tau(n) = \begin{cases} 2n & n \ (mod\ 2) = 0, \\ n+1 & n \ (mod\ 4) = 1, \\ \frac{n-1}{2} & n \ (mod\ 4) = 3. \end{cases}$$

In general:

Canonical arrows describe

case-by-case operations on modulo classes.

Making things unnecessarily complicated

Question: Does this diagram commute?

$$
\begin{tikzcd}
\mathbb{N} \arrow[d, "\tau"'] & & \mathbb{N} \arrow[ll, "\tau"'] \arrow[rr, "\tau^{-1}"] \arrow[d, "\tau^{-1}\star\tau"'] & & \mathbb{N} \arrow[d, "\tau^{-1}"] \\
\mathbb{N} & & \mathbb{N} \arrow[dl, "\tau"] \arrow[dr, "\tau^{-1}"'] & & \mathbb{N} \\
& \mathbb{N} \arrow[ul, "(\tau(\tau\star 1))\star 1"] & & \mathbb{N} \arrow[ur, "1\star(\tau^{-1}(1\star\tau))"'] &
\end{tikzcd}
$$

- **Category Theory:** Yes ... it's trivial (5 simple steps).
- **Direct Calculation:** Yes ... after a case-by-case analysis of 2^5 modulo classes

$$\{n \ (mod\ 32) = k\}_{k=0\ldots31}$$

Making things unnecessarily complicated

Question: Does this diagram commute?

$$\begin{tikzcd}
\mathbb{N} \arrow[d, "\tau"'] & & \mathbb{N} \arrow[ll, "\tau"'] \arrow[rr, "\tau^{-1}"] \arrow[d, "\tau^{-1}\star\tau"'] & & \mathbb{N} \arrow[d, "\tau^{-1}"] \\
\mathbb{N} & & \mathbb{N} \arrow[dl, "\tau"] \arrow[dr, "\tau^{-1}"'] & & \mathbb{N} \\
& \mathbb{N} \arrow[ul, "(\tau(\tau\star 1))\star 1"] & & \mathbb{N} \arrow[ur, "1\star(\tau^{-1}(1\star\tau))"'] &
\end{tikzcd}$$

- **Category Theory:** Yes ... it's trivial (5 simple steps).
- **Direct Calculation:** Yes ... after a case-by-case analysis of 2^5 modulo classes

$$\{n \ (mod\ 32) = k\}_{k=0\ldots31}$$

Making things unnecessarily complicated

Question: Does this diagram commute?

$$
\begin{tikzcd}
\mathbb{N} \arrow[d, "\tau"'] & & \mathbb{N} \arrow[ll, "\tau"'] \arrow[rr, "\tau^{-1}"] \arrow[d, "\tau^{-1}\star\tau"] & & \mathbb{N} \arrow[d, "\tau^{-1}"] \\
\mathbb{N} & & \mathbb{N} \arrow[dl, "\tau"] \arrow[dr, "\tau^{-1}"'] & & \mathbb{N} \\
& \mathbb{N} \arrow[ul, "(\tau(\tau\star1))\star1"] & & \mathbb{N} \arrow[ur, "1\star(\tau^{-1}(1\star\tau))"'] &
\end{tikzcd}
$$

- **Category Theory:** Yes ... it's trivial (5 simple steps).
- **Direct Calculation:** Yes ... after a case-by-case analysis of 2^5 modulo classes

$$\{n \ (mod\ 32) = k\}_{k=0\ldots31}$$

A couple of (semi-open) questions:

1. Is this telling us something concrete about complexity classes?
2. Where do we find such arithmetic operations used 'in the wild' ?

$$\tau(n) = \begin{cases} 2n & n \ (mod\ 2) = 0, \\ n+1 & n \ (mod\ 4) = 1, \\ \frac{n-1}{2} & n \ (mod\ 4) = 3. \end{cases}$$

A closely related question

What is the complexity of the word problem for Thompson's group $\mathcal{F}$?

$$\mathcal{F} = \langle x_0, x_1, x_2, \ldots | x_{n-a}^{-1} x_n x_{n-a} = x_{n+1} \text{ for } a > 0 \rangle$$

Gersten (1991) The Dehn function is at most exponential.

Gersten (1991) (Conjecture) It is precisely exponential!

Various (1991-2002) The bound slowly drops: $O(n^5)$, $O(n^{2.746})$, ...

(Guba 2002) The Dehn function is **quadratic**, $O(n^2)$.

What is the complexity of the word problem for Thompson's group $\mathcal{F}$?

$$\mathcal{F} = \langle x_0, x_1, x_2, \ldots | x_{n-a}^{-1} x_n x_{n-a} = x_{n+1} \text{ for } a > 0 \rangle$$

Gersten (1991) The Dehn function is at most exponential.

Gersten (1991) (Conjecture) It is precisely exponential!

Various (1991-2002) The bound slowly drops: $O(n^5)$, $O(n^{2.746})$, ...

(Guba 2002) The Dehn function is **quadratic**, $O(n^2)$.

A closely related question

What is the complexity of the word problem for Thompson's group $\mathcal{F}$?

$$\mathcal{F} = \langle x_0, x_1, x_2, \ldots | x_{n-a}^{-1} x_n x_{n-a} = x_{n+1} \text{ for } a > 0 \rangle$$

Gersten (1991) The Dehn function is at most exponential.

Gersten (1991) (Conjecture) It is precisely exponential!

Various (1991-2002) The bound slowly drops: $O(n^5)$, $O(n^{2.746})$, ...

(Guba 2002) The Dehn function is **quadratic**, $O(n^2)$.

A closely related question

What is the complexity of the word problem for Thompson's group $\mathcal{F}$?

$$\mathcal{F} = \langle x_0, x_1, x_2, \ldots | x_{n-a}^{-1} x_n x_{n-a} = x_{n+1} \ \text{for } a > 0 \rangle$$

Gersten (1991) The Dehn function is at most exponential.

Gersten (1991) (Conjecture) It is precisely exponential!

Various (1991-2002) The bound slowly drops: $O(n^5)$, $O(n^{2.746})$, ...

(Guba 2002) The Dehn function is **quadratic**, $O(n^2)$.

A closely related question

What is the complexity of the word problem for Thompson's group $\mathcal{F}$?

$$\mathcal{F} = \langle x_0, x_1, x_2, \ldots | x_{n-a}^{-1} x_n x_{n-a} = x_{n+1} \ \text{ for } a > 0 \rangle$$

Gersten (1991) The Dehn function is at most exponential.

Gersten (1991) (Conjecture) It is precisely exponential!

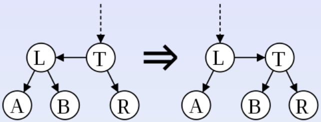
Various (1991-2002) The bound slowly drops: $O(n^5)$, $O(n^{2.746})$, ...

(Guba 2002) The Dehn function is **quadratic**, $O(n^2)$.

A relevant result

From a group theory textbook ...

Thompson's group is generated by rearrangements of the form:



Mark V. Lawson (2006)

In any single-object monoidal category $(\mathcal{C}, \star, \tau)$,

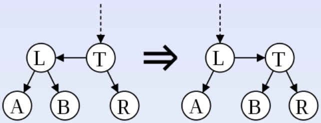
The arrows $\tau, (1 \star \tau)$ generate a copy of $\mathcal{F}$.

The group of canonical isomorphisms contains $\mathcal{F}$ as a proper subgroup.

A relevant result

From a group theory textbook ...

Thompson's group is generated by rearrangements of the form:



Mark V. Lawson (2006)

In any single-object monoidal category $(\mathcal{C}, \star, \tau)$,

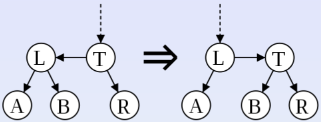
The arrows $\tau, (1 \star \tau)$ generate a copy of $\mathcal{F}$.

The group of canonical isomorphisms contains $\mathcal{F}$ as a proper subgroup.

A relevant result

From a group theory textbook ...

Thompson's group is generated by rearrangements of the form:



Mark V. Lawson (2006)

In any single-object monoidal category $(\mathcal{C}, \star, \tau)$,

The arrows $\tau, (1 \star \tau)$ generate a copy of $\mathcal{F}$.

The group of canonical isomorphisms contains $\mathcal{F}$ as a proper subgroup.

Using the ‘Cantor tensor’

The following two bijections generate a copy of Thompson’s $\mathcal{F}$.

$$A(n) = \begin{cases} 2n & n \ (mod\ 2) = 0, \\ n+1 & n \ (mod\ 4) = 1, \\ \frac{n-1}{2} & n \ (mod\ 4) = 3. \end{cases}$$

$$B(n) = \begin{cases} n & n \ (mod\ 2) \ = \ 0, \\ 2n-1 & n \ (mod\ 4) \ = \ 1, \\ n+2 & n \ (mod\ 8) \ = \ 3, \\ \frac{n-1}{2} & n \ (mod\ 8) \ = \ 7. \end{cases}$$

Arithmetic as category theory

Order-preserving bijections $\mathbb{N} \uplus \mathbb{N} \cong \mathbb{N}$

- are in 1:1 correspondence with points of the Cantor set[2].
- each determine a distinct tensor & associativity iso. on $\mathbb{N}$

In each case:

We derive a distinct representation of Thompson's group.

This is a good way of confusing group theorists!

[2] excluding a subset of measure zero.

Just give us time to work it out!

Every division of $\mathbb{N}$ into two infinite subsets determines such a bijection $\mathbb{N} \uplus \mathbb{N} \ \cong \ \mathbb{N}$.

N odd.	N even	*Trivial!*
$N \ (mod \ k) = 0$	$N \ (mod \ k) \neq 0$	*simple ...*
$N = p^n$	$N \neq p^n$	*interesting ...*
N prime	N non-prime	*complicated ...*
Statement with Gödel number N is provable.	Statement with Gödel number N is not provable.	*Subtle!*

Where else do we see associativity isomorphisms?

In which other settings might we find:

For $n \in \mathbb{N}$,

$$\tau(n) = \begin{cases} 2n & n \in 2\mathbb{N}, \\ n+1 & n \in 4\mathbb{N}+1, \\ \frac{n-1}{2} & n \in 4\mathbb{N}+3. \end{cases}$$

or similar untyped associativity isomorphisms?

Going round in circles

At least one interesting setting:

For $n \in \mathbb{Z}_p$,

$$\tau(n) = \begin{cases} 2n & n \in A, \\ n+1 & n \in B, \\ \frac{1}{2}(n-1) & n \in C. \end{cases}$$

where $\mathbb{Z}_p = A \uplus B \uplus C$.

That's all, folks (!)

Coming up next time ...

What all this has to do with:

- The Cantor space.
- Shuffling decks of cards.
- Young tableaux.
- Inverse semigroup theory.
- Linear logic & state machines.
- Some more modular arithmetic.