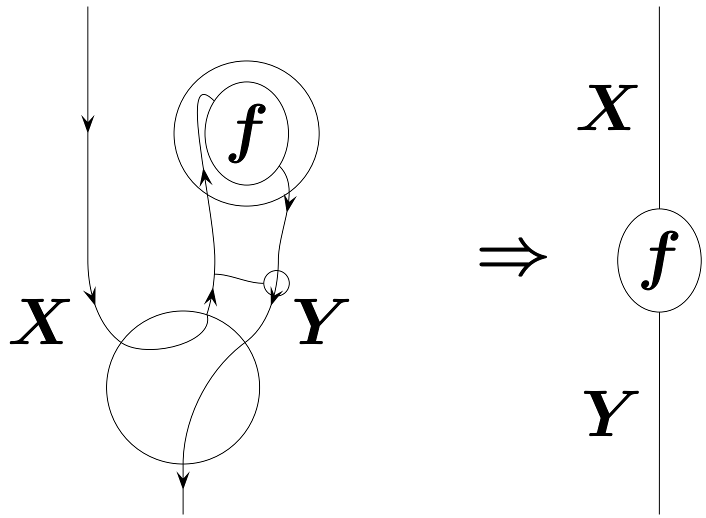
Computation and the Periodic Table

John C. Baez

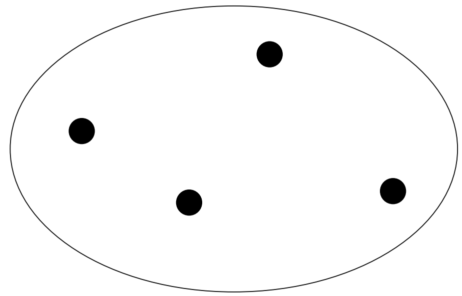
ATMCS 2008



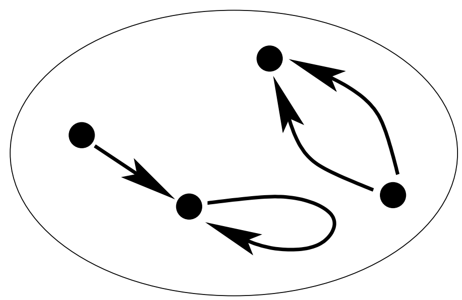
$$(\lambda x\!:\!X.f(x))(a) \quad \Rightarrow \quad f(a)$$

for references and more, see
http://math.ucr.edu/home/baez/periodic/

Once upon a time, mathematics was all about *sets*:

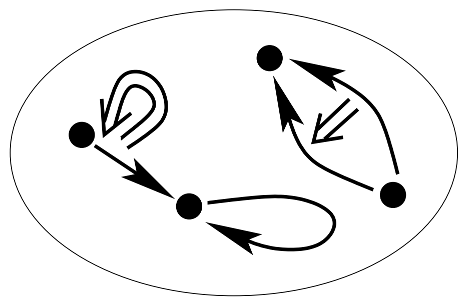


In 1945, Eilenberg and Mac Lane introduced *categories*:

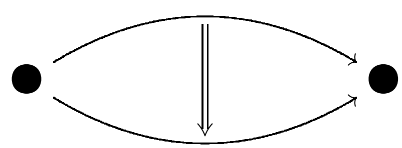


Category theory puts *processes* (morphisms): $\bullet \longrightarrow \bullet$
on an equal footing with *things* (objects): $\bullet$

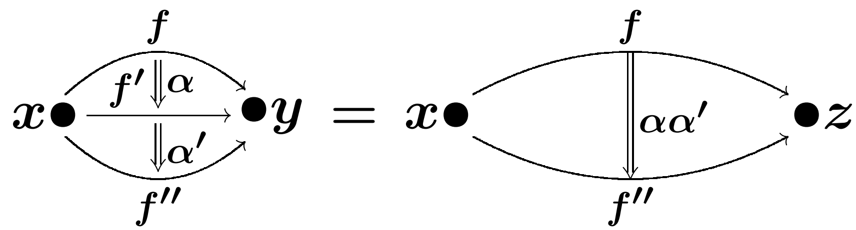
In 1967 Bénabou introduced *weak 2-categories*:



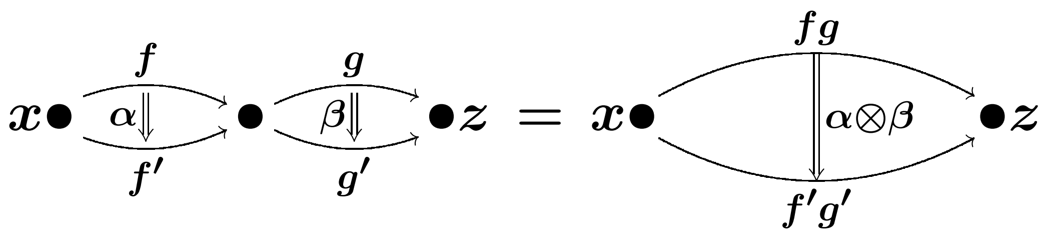
These include *processes between processes*, or '2-morphisms':



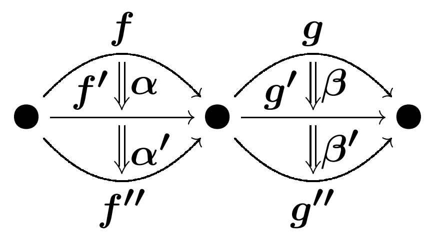
We can compose 2-morphisms vertically:



or horizontally:



and various laws hold, including the 'interchange' law:

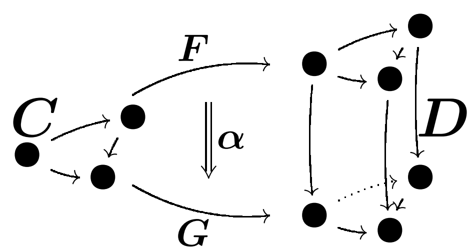


$$(\alpha\alpha') \otimes (\beta\beta') = (\alpha \otimes \beta)(\alpha' \otimes \beta')$$

The 'set of all sets' is really a category: Set.

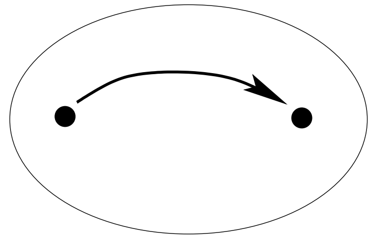
The 'category of all categories' is really a 2-category: Cat. It has:

- **categories as objects,**
- **functors as morphisms,**
- **natural transformations as 2-morphisms.**

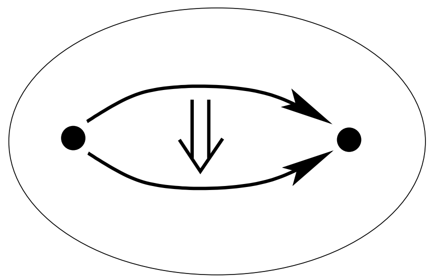


Cat is a 'strict' 2-category: all laws hold *exactly*, not just up to isomorphism. But there are also many interesting *weak* 2-categories!

For example, any topological space has a *fundamental groupoid*:



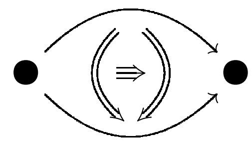
It also has a *fundamental 2-groupoid*:



This is a weak 2-category with:

- **points as objects,**
- **paths as morphisms,**
- **homotopy classes of 'paths of paths' as 2-morphisms.**

In 1995, Gordon, Power and Street introduced *weak 3-categories*:



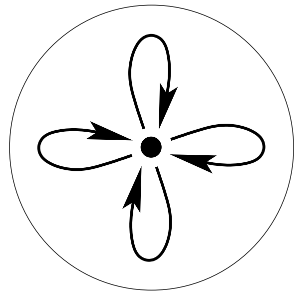
Now people are studying *weak n-categories* and even ∞-categories. This is starting to have a big impact on topology and physics. How about computation?

Is computation about processes between processes between processes...?

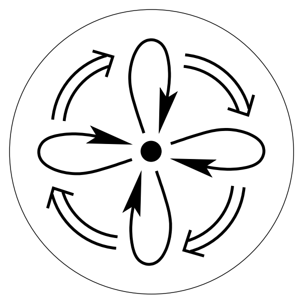
***Yes!* But to orient ourselves, we need some hypotheses about how n-categories work.**

My favorite is the 'Periodic Table'.

A category with one object is a *monoid* — a set with associative multiplication and a unit element:



A 2-category with one object is a *monoidal category* — a category with an associative 'tensor product' and a unit object:



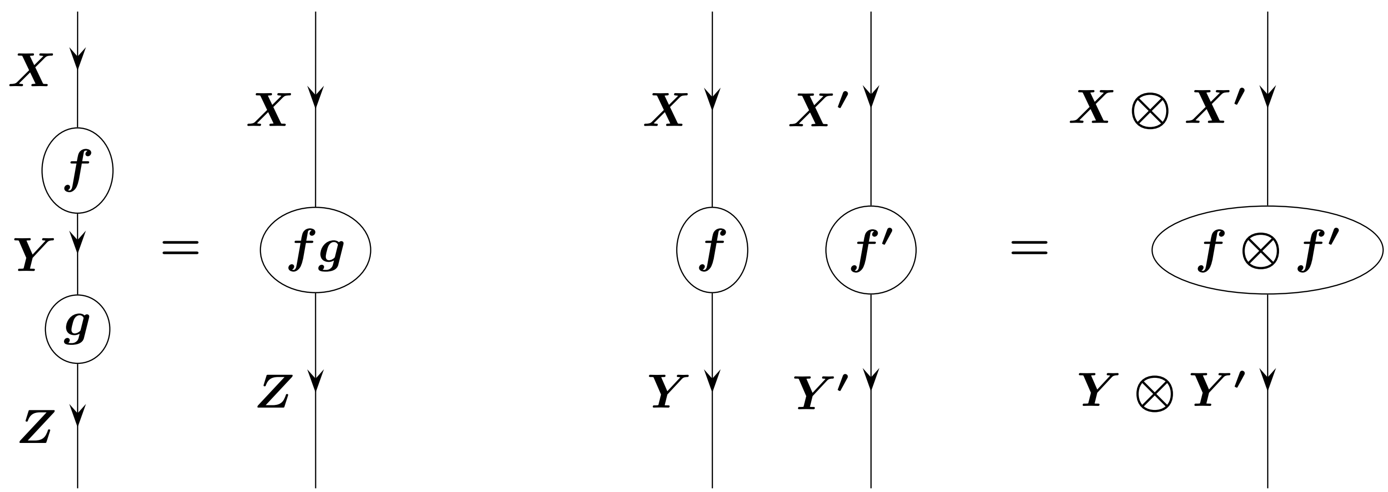
Now associativity and the unit laws are 'weakened':

$$(x \otimes y) \otimes z \cong x \otimes (y \otimes z), \qquad I \otimes x \cong x \cong x \otimes I$$

To regard a 2-category with one object as a monoidal category:

- **we ignore the object,**
- **we rename the morphisms 'objects',**
- **we rename the 2-morphisms 'morphisms'.**

Vertical and horizontal composition of 2-morphisms become composition and tensoring of morphisms:



In general, we expect an n-category with one object is a *monoidal* $(n-1)$*-category*.

For example:

- **Set is a monoidal category, using the cartesian product $S \times T$ of sets.**
- **Cat is a monoidal 2-category, using the cartesian product $C \times D$ of categories.**
- **We expect that nCat is a monoidal $(n+1)$-category!**

QUESTION: what's a *monoidal category* with just one object? It must be some sort of monoid...

It has one object, namely the unit I, and a set of morphisms $\alpha\colon I \to I$. We can compose morphisms:

$$\alpha\beta$$

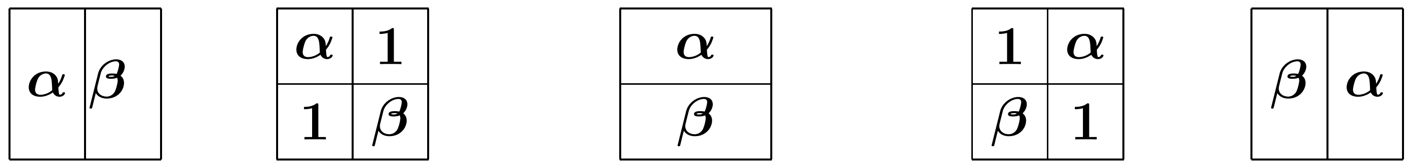
and also tensor them:

$$\alpha \otimes \beta$$

Composition and tensoring are related by the interchange law:

$$(\alpha\alpha') \otimes (\beta\beta') = (\alpha \otimes \beta)(\alpha' \otimes \beta')$$

So, we can carry out the 'Eckmann–Hilton argument':



$$\begin{array}{ccccc} \alpha \otimes \beta & = (\alpha \otimes 1)(1 \otimes \beta) & & (1\beta) \otimes (\alpha 1) = & \beta \otimes \alpha \\ & \| & & \| & \\ & (\alpha 1) \otimes (1\beta) \;=\; & \alpha\beta & =\; (1 \otimes \alpha)(\beta \otimes 1) & \end{array}$$

ANSWER: a monoidal category with one object is a *commutative* monoid!

In other words: a 2-category with one object and one morphism is a commutative monoid.

What's the pattern?

An $(n+k)$-category with only one j-morphism for $j < k$ can be reinterpreted as an n-category.

But, it will be an n-category with k ways to 'multiply': a *k-tuply monoidal n-category*.

For example $n = 1, k = 1$: a 2-category with one object is a monoidal category.

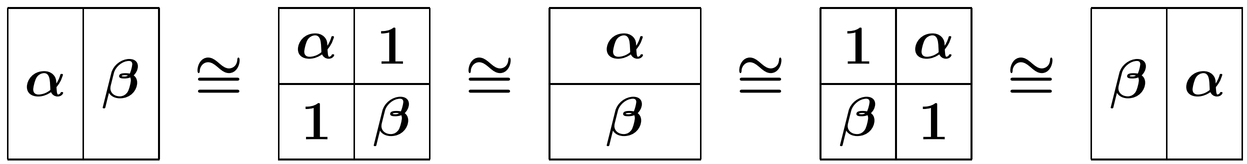
When there are several ways to multiply, the Eckmann–Hilton argument gives a kind of 'commutativity'.

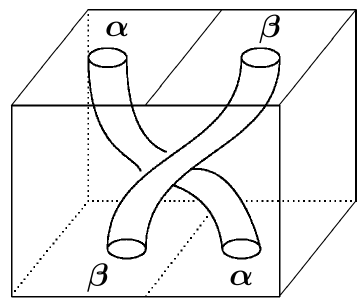
Our guesses are shown in the Periodic Table...

k-tuply monoidal n-categories

	$n = 0$	$n = 1$	$n = 2$
$k = 0$	sets	categories	2-categories
$k = 1$	monoids	monoidal categories	monoidal 2-categories
$k = 2$	commutative monoids	braided monoidal categories	braided monoidal 2-categories
$k = 3$	‘’	symmetric monoidal categories	sylleptic monoidal 2-categories
$k = 4$	‘’	‘’	symmetric monoidal 2-categories
$k = 5$	‘’	‘’	‘’

Consider $n = 1$, $k = 2$: a doubly monoidal 1-category is a *braided monoidal category*. The Eckmann–Hilton argument gives the braiding:

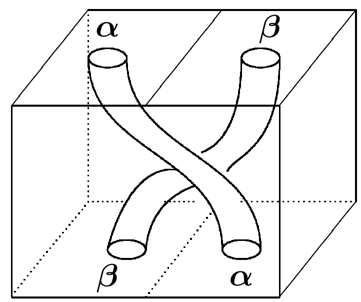




$$B_{\alpha,\beta}\colon \alpha \otimes \beta \xrightarrow{\sim} \beta \otimes \alpha$$

The *process of proving an equation* has become an *isomorphism!* This happens when we move one step right in the Periodic Table.

Indeed, a *different proof* of commutativity becomes a *different isomorphism*:

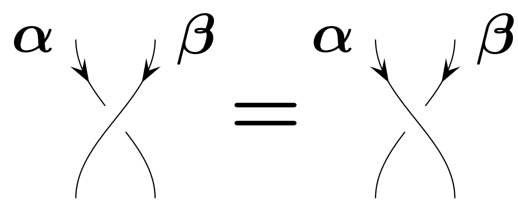


$$B^{-1}_{\beta,\alpha} \colon \alpha \otimes \beta \xrightarrow{\sim} \beta \otimes \alpha$$

This explains the existence of knots!

Shum's theorem: $1\mathrm{Tang}_2$, the category of 1d tangles in a (1+2)-dimensional cube, is the free braided monoidal category with duals on one object.

A triply monoidal 1-category is a *symmetric monoidal category*. Now we have 'three dimensions of space' instead of just two. This makes the two ways of moving α past β equal:



So, the situation is 'more commutative'. This happens when we move one step down in the Periodic Table.

We can untie all knots in 4d:

Theorem: $1\mathrm{Tang}_3$, the category of 1d tangles in a $(1+3)$-dimensional cube, is the free symmetric monoidal category with duals on one object.

However, k-tuply monoidal n-categories seem to become 'maximally commutative' when k reaches $n+2$.

For example, you can untie all n-dimensional knots in a $(2n + 2)$-dimensional cube. Extra dimensions don't help!

Stabilization Hypothesis: k-tuply monoidal n-categories are equivalent to $(k + 1)$-tuply monoidal n-categories when $k \geq n + 2$.

We call these *stable n-categories*. These should serve as abstract contexts for computation in which data doesn't get 'tangled up' as it moves.

nCat should be a stable $(n + 1)$-category.

Now, what about computation?

Topological quantum computation uses *braided* monoidal categories, but more often we use *symmetric* monoidal categories where:

- objects are *types* $X, Y, Z, \ldots$
- morphisms $f \colon X \to Y$ are equivalence classes of *terms* of type Y with free variable of type X.

For example: Lambek showed that any theory in the simply typed λ-calculus gives a cartesian closed category. Two terms give the same morphism if they differ by certain *rewrite rules*, such as β-reduction:

$$(\lambda x : X. f(x))(a) \;\Rightarrow\; f(a)$$

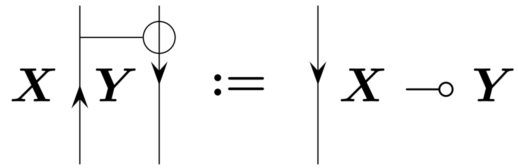
Identifying terms that differ by rewrite rules amounts to *ignoring the process of computation!* To avoid this, use a 2-category where:

- objects are *types* $X, Y, Z, \ldots$
- morphisms $f\colon X \to Y$ are *terms* of type Y with free variable of type X.
- 2-morphisms $\alpha\colon f \Rightarrow g$ are equivalence classes of *sequences of rewrites* going from f to g.

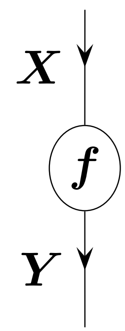
Any theory in the simply-typed λ-calculus should give a 'cartesian closed 2-category' this way.

More generally, we should get 'monoidal closed 2-categories' where the 2-morphisms are *processes of computation*.

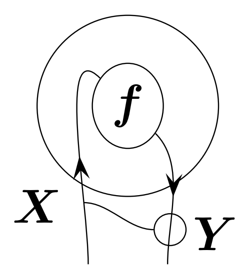
In a monoidal closed 2-category, any pair of objects (types) X, Y has a 'function type' $X \multimap Y$:



Any morphism $f\colon X \to Y$:

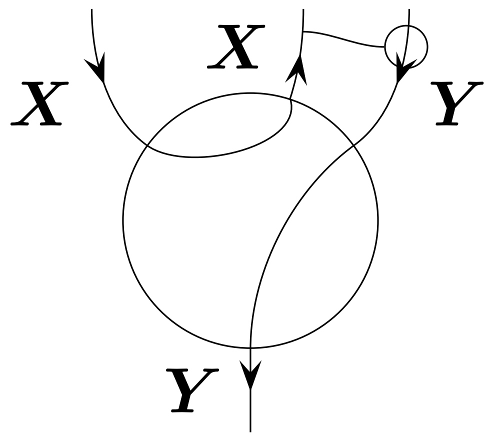


has a 'name' $\ulcorner f \urcorner\colon I \to (X \multimap Y)$:

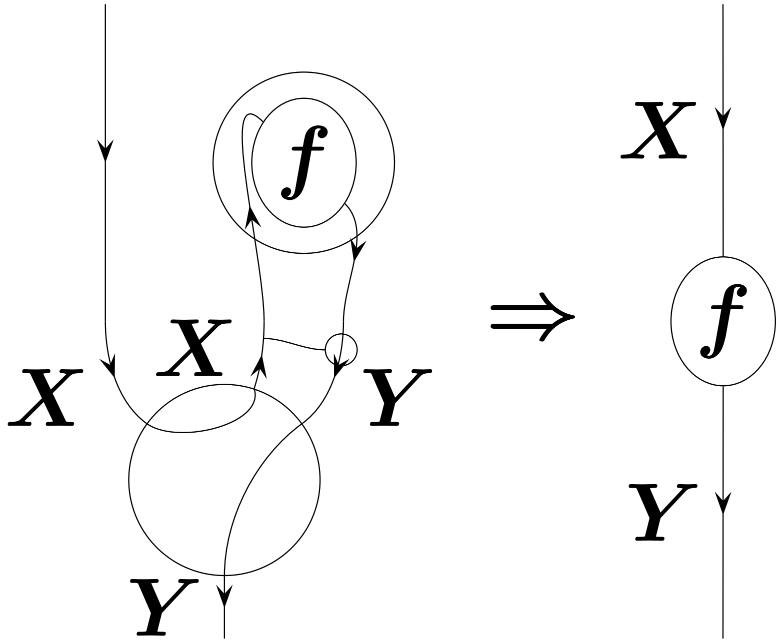


We also have an 'evaluation' morphism:

$$\mathrm{ev}_{X,Y} \colon X \otimes (X \multimap Y) \to Y$$



But, evaluating the name of f does not give f. It gives a morphism *isomorphic* to f via some 2-morphism:

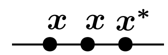


In the λ-calculus, this 2-morphism corresponds to β-reduction:

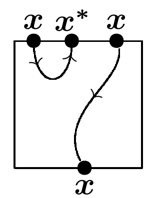
$$(\lambda x : X . f(x))(a) \;\Rightarrow\; f(a)$$

This 2-morphism exists in any monoidal closed 2-category. For example $2\mathrm{Tang}_1$, which has:

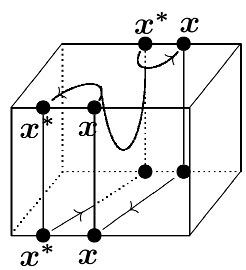
- **collections of oriented points in the 1-cube as objects:**



- **1d tangles in the 2-cube as morphisms:**

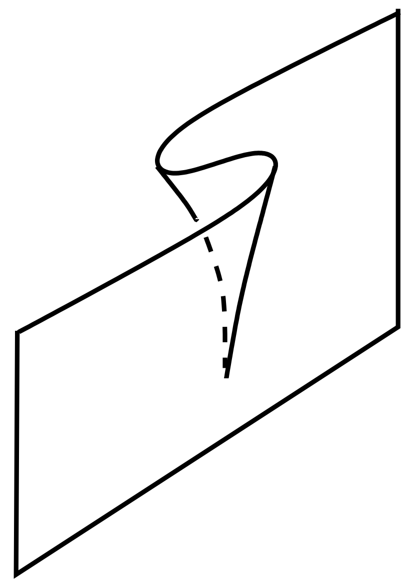


- **isotopy classes of 2d tangles in the 3-cube as 2-morphisms:**



Tangle Hypothesis: $n\mathrm{Tang}_k$ is the free k-tuply monoidal n-category with duals on one object.

The 2-morphism analogous to β-reduction in $\mathbf{2Tang}_1$ is the *fold catastrophe*:



Like β-reduction, it 'straightens out a zig-zag'.

This is the beginning of a long, unfinished story relating computation, topology and the Periodic Table.

k-tuply monoidal n-categories

	$n = 0$	$n = 1$	$n = 2$
$k = 0$	sets	categories	2-categories
$k = 1$	monoids	monoidal categories	monoidal 2-categories
$k = 2$	commutative monoids	braided monoidal categories	braided monoidal 2-categories
$k = 3$	‘’	symmetric monoidal categories	sylleptic monoidal 2-categories
$k = 4$	‘’	‘’	symmetric monoidal 2-categories
$k = 5$	‘’	‘’	‘’

See my webpage for links to references, e.g.:

- R. A. G. Seely, Modeling computations in a 2-categorical framework, *LICS* 1987.
- Barnaby P. Hilken, Towards a proof theory of rewriting: the simply-typed 2λ-calculus, *Theor. Comp. Sci.* 170 (1996), 407.
- Albert Burroni, Higher-dimensional word problems with applications to equational logic, *Theor. Comp. Sci.* 115 (1993), 43.
- Yves Guiraud, The three dimensions of proofs, *Ann. Pure Appl. Logic* 141 (2006), 266.
- Vladimir Voevodsky, A very short note on the homotopy lambda calculus, 2006.

and also my seminar and work with Mike Stay.