QUANTUM GRAVITY SEMINAR
Spring 2004

30 March 2004

Properties, Structures, 'n' Stuff!

Mathematical gadgets are often defined by:

specifying stuff (e.g. a set, several sets, etc.

equipped with structure (e.g. functions, elements, relations, collections of subsets)

satisfying certain properties (e.g. equations, inequalities, inclusions)

E.g.: a function is:

a pair of sets X, Y

equipped with $f \subseteq X \times Y$

satisfying $\forall x \in X \ \exists!\, y \in Y$ s.t. $(x,y) \in f$

You can check properties: they're either true or false.

You can choose structures from among a set of possibilities.

You can choose stuff from among a category of possibilities.

(But each step depends on the following ones.)

Note: Here's a lexicon for Logicians:

types .... stuff
predicates .... structure
axioms .... properties

We see a kind of hierarchy here:

properties	$\{F,T\}$	=	the 0-category of all -1-categories
structures	Set	=	the 1-category of all 0-categories
stuff	Cat	=	the 2-category of all categories

where: a 2-category is a 2-category
a 1-category is a category
a 0-category is a set
& a -1-category is a truth value

So there's a pattern here (which can be extended, but we won't talk about that now).

There are some subtleties, e.g.: sometimes structure can be reinterpreted as properties. E.g. we can define a monoid as:

stuff → • a set M equipped with
structure → $\cdot : M \times M \longrightarrow M$ & $1 \in M$
properties → s.t. $\forall x, y, z \in M$ $(xy)z = x(yz)$ & $1x = x = x1$

or

stuff → a set M equipped with
structure → $\cdot : M \times M \to M$
properties → s.t. $\forall x, y, z \in M$, $(xy)z = x(yz)$
& $\exists x \in M \ \forall y \in M \quad xy = y = yx$
(note this x is automatically unique!)

These definitions give the same notion of monoid, but different notions of a morphism between monoids — hence a different category of monoids. Morphisms between mathematical gadgets are:

- maps from stuff to corresponding stuff s.t.
- the structure is preserved

(Properties don't enter the def. of morphisms at all.)

For the first def of monoid, a morphism is:

- a function $f: M \to M'$
- preserving multiplication & unit:

$$f(xy) = f(x)\,f(y)$$
$$f(1) = 1'$$

For the second def, we get:

- a function $f: M \to M'$
- preserving multiplication

$$f(xy) = f(x)\,f(y)$$

These are really different, but they're the same in the case of isomorphisms. Let's see why this is true. If f is an isomorphism of the first sort, it's obviously one of the second sort, but the converse holds too: if $f: M \xrightarrow{\sim} M'$ preserves multiplication, it preserves the unit too:

$$f(1) = 1' \ ?$$

$$\Updownarrow$$

$$\forall y \in M', \quad f(1)y = y = y f(1) \ ?$$

$$\Updownarrow$$

$$\forall y \in M' \quad f^{-1}(f(1)y) = f^{-1}(y) = f^{-1}(y f(1)) \ ?$$

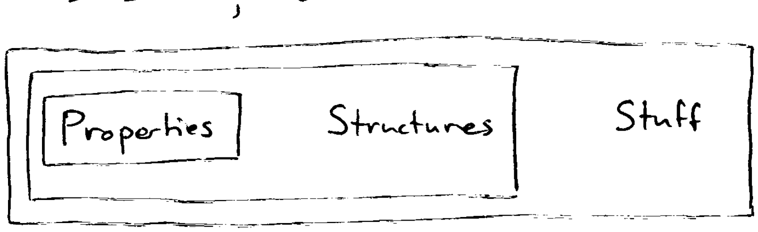
$$\Updownarrow$$

$$\forall y \in M' \quad 1 f^{-1}(y) = f^{-1}(y) = f^{-1}(y) 1 \ ?$$

$$\Updownarrow$$

$$\forall z \in M \quad 1z = z = z1 \quad \checkmark$$

The moral: whether we count something as structure or property (when we have this choice) doesn't affect the <u>iso</u>morphisms, so the <u>groupoid</u> of mathematical gadgets is more robust than the <u>category</u>. Similarly for stuff that can be reinterpreted as structure. (E.g. the unit of a monoid could be thought of as <u>stuff</u>: a 1-elt set $\{*\}$ with function $f: \{*\} \to M$) We can always reinterpret properties as structure & structure as stuff, but not vice versa:



1 April 2004

Properties, Structure & Stuff (continued)

We'll make these precise by considering what it means for a functor $F : C \longrightarrow D$ to forget nothing, properties, structure, or stuff.

Examples:

$$\begin{tikzcd} \mathrm{Set}^2 \arrow[d, "F"'] & (x,y) \arrow[d, mapsto] \\ \mathrm{Set} & x \end{tikzcd}$$

forgets stuff;

$$\begin{tikzcd} \mathrm{CommRing} \arrow[d, "F"'] & (R,+,\cdot,0,1) \arrow[d, mapsto] \\ \mathrm{Set} & R \end{tikzcd}$$

forgets structure;

$$\begin{tikzcd} \mathrm{CommRing} \arrow[d] & (R,+,\cdot,0,1) \arrow[d, mapsto] \\ \mathrm{Ring} & (R,+,\cdot,0,1) \end{tikzcd}$$

forgets properties.

$$\begin{tikzcd} \mathrm{Set} \arrow[d] & S \arrow[d, mapsto] \\ \mathrm{Set} & S \end{tikzcd}$$

forgets nothing

But also

$$\begin{tikzcd}
{[\mathbb{R}^n\text{'s, linear operators}]} \arrow[d, hook, "F"'] \\
\mathrm{FinVect}_{\mathbb{R}}
\end{tikzcd}$$

forgets nothing.

Now for the actual definitions... A set is "the same as" a category with only identity morphisms, where these identity morphisms are usually called "equations." i.e. $1_x : x \longrightarrow x$ is another name for $x = x$. A category is "the same as" a 2-category with only identity 2-morphisms; which are usually called equations between morphisms: $1_f : f \Rightarrow f$ is usually called $f = f$. So a set can be thought of as an ∞-category with:

objects	(elements)
identity morphisms	(equations between elts)
identity 2-morphisms	(equations between equations between elts.)
identity 3-morphisms	
⋮	

Similarly, a category is an ω-cat with:

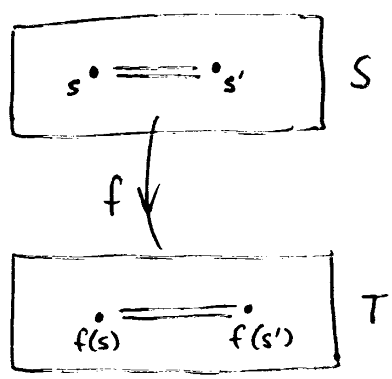
objects
morphisms
identity 2-morphisms (equations between morphisms)
identity 3-morphisms (equations between eqns between morphisms)

A famous little theorem: a function between sets is an isomorphism (i.e. has an inverse) iff it's

surjective = "onto for elements"

injective = "onto for equations"

Namely, $f: S \to T$ is surjective if for every element $t \in T$ there is some $\tilde{t} \in S$ with $f(\tilde{t}) = t$. It's injective if: given $f(s), f(s') \in T$, if $f(s) = f(s')$ then $s = s'$!



Recall that a functor $F: C \to D$ is an <u>equivalence</u> if it has an inverse up to natural isomorphism, i.e. $\exists\ G: D \to C$ s.t. $FG \cong 1 \cong GF$.

A bigger theorem: a functor between categories is an equivalence iff it's

essentially surjective	=	weakly onto for objects
full	=	onto for morphisms
faithful	=	onto for equations between morphisms (i.e. for commutative diagrams)

Namely, $F: C \to D$ is <u>essentially surjective</u> if for every object $d \in D$ there is some $\tilde{d} \in C$ with $F(\tilde{d}) \cong d$. It's <u>full</u> if given $F(c), F(c') \in D$ & given $f: F(c) \to F(c')$ then $\exists\ \tilde{f}: c \to c'$ s.t. $F(\tilde{f}) = f$. It's <u>faithful</u> if given $F(c), F(c') \in D$ & given $F(f), F(f'): F(c) \to F(c')$, if $F(f) = F(f')$ then $f = f'$. We normally say F is <u>full</u> if

$$F: \hom(c,c') \longrightarrow \hom(F(c), F(c'))$$

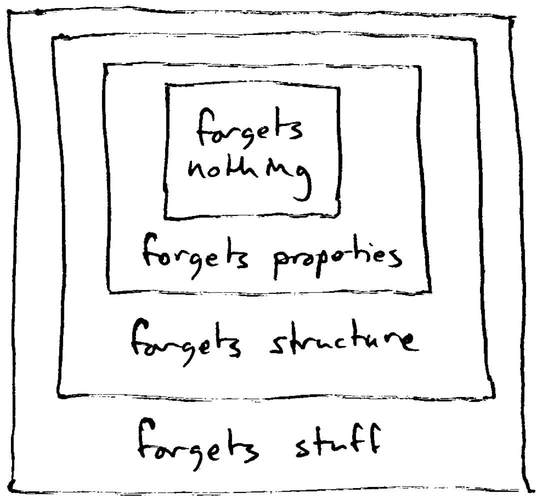
is onto, & faithful if this is one-to-one.

Now we say a functor:

- <u>forgets nothing</u> if it is ess. surj., full, and faithful (i.e. an equivalence)
- <u>forgets properties</u> if it's full & faithful
- <u>forgets structure</u> if it's faithful
- <u>forgets stuff</u> ... always!

(Note this gives <u>nested</u> definitions, so that each is a special case of the next. We could also make <u>partitioned</u> definitions - we'll do this later)

This gives these inclusions



Example:

$$\begin{array}{c} Set^2 \\ \big\downarrow F \\ Set \end{array}$$

as before.

This is essentially surjective, full, but <u>not</u> faithful since we could have $(f, g) \neq (f, g')$ but $f = f$