Index of the slides:

2. Program (1)

(Original draft, oct/2011...)

(1) Categorias como "estrutura" (objetos, Hom, identidade, composição) e "propriedades" (a composição é associativa, identidades se comportam como deveriam). Comparação com grupos, anéis, etc; "dependent types".

(2) Nossas categorias preferidas: Set, categorias finitas dadas explicitamente, posets, categorias de grupos, anéis, espaços topológicos, categorias de subobjetos.

(3) Funtores e transformações naturais. Diagramas e condições de coerência. Introdução ao lambda-cálculo tipado (em Set). Notação lambda para morfismos. Beta-redução (para composição). Produtos de objetos. Produtos como "estrutura" + "propriedades". Prova de que dois produtos P e P' são isomorfos. Como deixar em segundo plano a prova de que as compostas P->P'->P e P'->P->P' são identidades. Como isolar as "construções". Análise de provas do MacLane, do Awodey e do Beck.

(4) Categorias cartesianas. A linguagem interna de uma categoria cartesiana: uplas e projeções. Tradução entre linguagem interna e linguagem categórica.

(5) Uma categoria cartesiana abstrata: a CC livre gerada pelos objetos A_1,...,A_n. A definição formal (categórica) de "livre".

(6) Os funtores (×B) e (B->) em Set. A adjunção (×B)-|(B->). Adjunções como "estrutura" e "propriedades". Proto-adjunções (só a parte "estrutura" das adjunções). Unidades e counidades. O teorema sobre vários modos equivalentes de especificar uma adjunção. Categorias cartesianas fechadas ("CCC"s). A linguagem interna de uma CCC. Tradução entre linguagem interna e linguagem categórica. Em que sentido esta tradução é "fiel".

3. Program (2)

(7) Adjunções em posets. Como calcular "P->Q" num certo poset. Posets como espaços de valores de verdade. Cálculo proposicional num poset. Como calcular a "coimplicação" neste poset. A relação entre adjunções e produtos.

(8) A tradução entre cálculo de seqüentes, dedução natural e operações numa CCC. Uma tradução correspondente para lambda-cálculo. Curry-Howard. Reduções em demonstrações em dedução natural.

(9) Outros posets nos quais podemos fazer cálculo proposicional (intuicionista). Categorias de funtores. Categorias da forma 2^D, onde D é um poset finito "plano". Representação gráfica explícita dos objetos destas categorias. Valores de verdade como subobjetos. Lógica modal (S4) nestas categorias. "Necessário" como "interior". Cálculo proposicional num espaço topológico qualquer.

(10) Introdução à noção de "feixe": coberturas, colagens, duas noções de saturação. Categorias da forma Set^D ("prefeixes"). Exemplos de prefeixes que são e que não são feixes.

(11) O classificador de subobjetos em categorias da forma Set^D. Introdução à linguagem interna de um topos.

(12) O lema de Yoneda (versão "proto"). Funtores representáveis.

(13) Mônadas e álgebras. As categorias de Kleisli e Eilenberg-Moore de uma mônada. O teorema da comparação (versão "proto").

(14) Mônadas em linguagens funcionais.

(15) Morfismos geométricos num caso particular. Feixificação num caso particular.

4. Functions, explicitly

The two functions below,

$$\begin{array}{rcccc} f & : & \mathbb{N} & \to & \mathbb{N} \\ & & n & \mapsto & 3+n \end{array}$$

and

$$\begin{array}{rcccc} g & : & \mathbb{N} & \to & \mathbb{N} \\ & & n & \mapsto & 1+(n+2) \end{array}$$

are mathematically the same function,
even though they are syntactically different
and they correspond to *different algorithms*...

Mathematicians tend to think that functions
are their graphs — i.e., sets of ordered pairs.

For example, if

$$\begin{array}{rcccc} h & : & \mathbb{N} & \to & \mathbb{N} \\ & & n & \mapsto & 10n \end{array}$$

Then $h = \{(2, 20), (3, 30)\}$.
We will often write this as $\begin{smallmatrix} 2\mapsto 20 \\ 3\mapsto 30 \end{smallmatrix}$
or as $\{\begin{smallmatrix} 2\mapsto 20 \\ 3\mapsto 30 \end{smallmatrix}\}$.

5. Functions that return functions

We will often have to deal with *functions that return functions*.
For example:

$$\begin{array}{ccccc} m & : & \{2,3\} & \to & \mathbb{N}^{\{10,100\}} \\ & & a & \mapsto & \left(\begin{array}{ccccc} m_a & : & \{10,100\} & \to & \mathbb{N} \\ & & b & \mapsto & ab \end{array} \right) \end{array}$$

Then $(m(2))(10) = m_2(10) = 2 \cdot 10 = 20$,
$m = \{(2, m_2), (3, m_3)\}$,
$m = \{(2, \{(10, 20), (100, 200)\}), (3, \{(10, 20), (100, 200)\})\}$.

Using the other notation,
$m = \left\{ \begin{array}{l} 2 \mapsto \{ \begin{array}{l} 10 \mapsto 20 \\ 100 \mapsto 200 \end{array} \} \\ 3 \mapsto \{ \begin{array}{l} 10 \mapsto 30 \\ 100 \mapsto 300 \end{array} \} \end{array} \right\}$

6. Groups as structure plus properties

A group G is (formally) a tuple, whose components are:
- an underlying set, usually also called G,
- a binary operation, $\cdot : G \times G \to G$,
- an identity element, $e \in G$,
- a unary operation, $(\cdot)^{-1} : G \to G$,
- the guarantee that e "behaves as expected",
- the guarantee that $\cdot$ "behaves as expected",
- the guarantee that $(\cdot)^{-1}$ "behaves as expected".

So: $G = (G, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$,
where:
$\mathsf{idLR} \equiv (\forall a \in G.\, ea = a) \wedge (\forall a \in G.\, ae = a)$,
$\mathsf{assoc} \equiv (\forall a \in G.\, \forall b \in G.\, \forall c \in G.\, a(bc) = (ab)c)$,
$\mathsf{invLR} \equiv (\forall a \in G.\, aa^{-1} = e) \wedge (\forall a \in G.\, a^{-1}a = e)$,

Let's clean this up...
$G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$,
where:
- $G_0 : \mathsf{Sets}$,
- $e : G_0$,
- $\cdot : G_0 \times G_0 \to G_0$,
- $(\cdot)^{-1} : G_0 \to G_0$,
- idLR is a proof of $(\forall a \in G_0.\, ea = a) \wedge (\forall a \in G_0.\, ae = a)$,
- assoc is a proof of $(\forall a \in G_0.\, \forall b \in G_0.\, \forall c \in G_0.\, a(bc) = (ab)c)$,
- invLR is a proof of $(\forall a \in G_0.\, aa^{-1} = e) \wedge (\forall a \in G_0.\, a^{-1}a = e)$.

More generally:
$G = (G_0, \cdot_G, e_G, (\cdot)_G^{-1}; \mathsf{idLR}_G, \mathsf{assoc}_G, \mathsf{invLR}_G)$.

7. Groups as structure plus properties (2)

$G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$,
where:
- $G_0 : \mathsf{Sets}$,
- $e : G_0$,
- $\cdot : G_0 \times G_0 \to G_0$,
- $(\cdot)^{-1} : G_0 \to G_0$,
- idLR is a proof of $(\forall a \in G_0.\, ea = a) \wedge (\forall a \in G_0.\, ae = a)$,
- assoc is a proof of $(\forall a \in G_0.\, \forall b \in G_0.\, \forall c \in G_0.\, a(bc) = (ab)c)$,
- invLR is a proof of $(\forall a \in G_0.\, aa^{-1} = e) \wedge (\forall a \in G_0.\, a^{-1}a = e)$.

Constructing (*exhibiting*) a group G means:
- *choosing* a set G_0,
- *choosing* an element $e : G_0$
- *choosing* a function $\cdot : G_0 \times G_0 \to G_0$
- *choosing* a function $(\cdot)^{-1} : G_0 \to G_0$,
- *proving* that $(\forall a \in G_0.\, ea = a) \wedge (\forall a \in G_0.\, ae = a)$,
- *proving* that $(\forall a \in G_0.\, \forall b \in G_0.\, \forall c \in G_0.\, a(bc) = (ab)c)$,
- *proving* that $(\forall a \in G_0.\, aa^{-1} = e) \wedge (\forall a \in G_0.\, a^{-1}a = e)$.

Note:
':' is roughly the same as '$\in$', so
$e : G_0$ is roughly the same as $e \in G_0$;
Sets is the class of all sets (the "set of all sets"), so
$G_0 : \mathsf{Sets}$ means that G_0 is a set;
$A \to B$ will be regarded as the set B^A, so
$f : A \to B$ means $f \in B^A$, i.e.,
f is an element of the set of all functions from A to B.

8. Groups as structure plus properties (3)

Fact 1: when we see the *name* of an object
we often know which *type* it is expected to have.
(We will not formalize this!)

Fact 2: it is possible to define
the set of proofs of $(\forall a \in G_0.\, ea = a) \wedge (\forall a \in G_0.\, ae = a)$,
and then say: $\mathsf{idLR} : \mathsf{Proofs}[(\forall a \in G_0.\, ea = a) \wedge (\forall a \in G_0.\, ae = a)]$.
(We will not formalize this, either.)

Fact 3: it is *sometimes* sufficient to say just this,
$G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$,
and our interlocutors will be able to figure out by themselves
the (implicit) types of all the components.

The first four components of the tuple
$G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$
will be called the "structure" part of G; their types
involve just known objects (the class Sets), and '$:$', '$\times$', '$\to$'.

Note that after '$G_0 : \mathsf{Sets}$' the value of G_0 is a "known object",
and the declaration '$\cdot : G_0 \times G_0 \to G_0$' makes sense.

The last three components of the tuple
$G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$
will be called the "properties" part of G; their types
involve 'Proofs', quantifiers, **equalities**, **equations**...

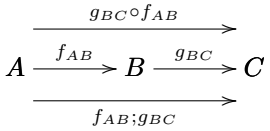
Conventions:
- a ';' separates the "structure" from the "properties"
- $G = (G_0, \cdot, e, (\cdot)^{-1}; \mathsf{idLR}, \mathsf{assoc}, \mathsf{invLR})$ is a group,
- $G^- = (G_0, \cdot, e, (\cdot)^{-1})$ is a **proto-group**;
- each definition of a structure will have a corresponding proto-structure.

9. Categories as structure plus properties

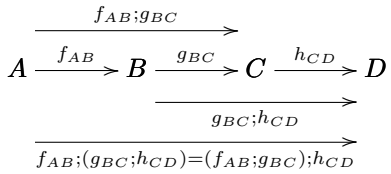
Informally, a *Category* is a directed graph
in which the arrows can be composed (when they "match"),
the composition is associative,
and we have identity arrows, that behave as expected.

A bit more formally:
$\mathbf{C} = \{\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \circ_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}; \mathsf{assoc}_{\mathbf{C}}, \mathsf{idLR}_{\mathbf{C}}\}$
where:
- $\mathbf{C}_0$ is the set of vertices,
- $\mathrm{Hom}_{\mathbf{C}}(A, B)$ is the set of arrows from A to B,
- for any $A, B, C \in \mathbf{C}_0$ we have

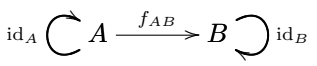
$$\begin{array}{rcccc} \circ_{\mathbf{C}\,ABC} & : & \mathrm{Hom}_{\mathbf{C}}(B,C) \times \mathrm{Hom}_{\mathbf{C}}(A,B) & \to & \mathrm{Hom}_{\mathbf{C}}(A,C) \\ & & (g_{BC},\ f_{AB}) & \mapsto & g_{BC} \circ f_{AB} \end{array}$$



- for any $A \in \mathbf{C}_0$ we have an "identity arrow" $\mathrm{id}_{\mathbf{C}}(A) \in \mathrm{Hom}_{\mathbf{C}}(A, B)$,
- for any $A, B, C, D \in \mathbf{C}_0$ and arrows f_{AB}, g_{BC}, h_{CD} we have:



- for any $A, B \in \mathbf{C}_0$ and f_{AB} we have $\mathrm{id}_A; f_{AB} = f_{AB} = f_{AB}; \mathrm{id}_B$.



10. Categories as structure plus properties (2)

$\mathbf{C} = \{\mathbf{C}_0, \mathrm{Hom}, \circ, \mathrm{id}; \mathsf{assoc}, \mathsf{idLR}\}$,
where:

- $\mathbf{C}_0 : \mathsf{Sets}$ (in our "small categories")
- $\mathrm{Hom} : \mathbf{C}_0 \times \mathbf{C}_0 \to \mathbf{Set}$
- for any A, B, C we have $\circ_{ABC} : \mathrm{Hom}(B, C) \to \mathrm{Hom}(A, B) \to \mathrm{Hom}(A, C)$
- for any A we have $\mathrm{id}(A) : \mathrm{Hom}(A, A)$
- assoc is a proof of

$$\forall A, B, C, D, f_{AB}, g_{BC}, h_{CD}.\, (f_{AB}; g_{BC}); h_{CD} = f_{AB}; (g_{BC}; h_{CD})$$

- idLR is a proof of $\forall A, B, f_{AB}.\, \mathrm{id}_A; f_{AB} = f_{AB} = f_{AB}; \mathrm{id}_B$

11. Set as a category

Our archetypical category is **Set**.
- $\mathbf{Set}_0 = \mathsf{Sets}$ (the objects of **Set** are sets),
- $\mathrm{Hom}_{\mathbf{Set}}(A, B)$ = the set of all functions from A to B,
- $f \circ_{\mathbf{Set}} g$ is the composition of functions, $f \circ g$
- $\mathrm{id}_{\mathbf{Set}}(A)$ is the identity map $\mathrm{id}_A : A \to A$.

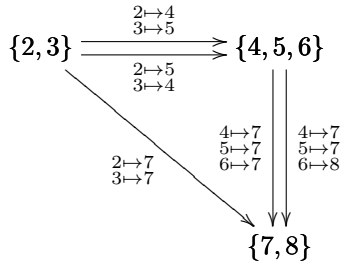
[EXERCISE: prove $\mathsf{assoc}_{\mathbf{Set}}$ and $\mathsf{idLR}_{\mathbf{Set}}$].

When we look at **Set** as a graph (with composition and ids), it is a HUGE graph — to many vertices, too many arrows. Let's see two simple ways to define smaller categories.

12. Finite subcategories of Set, explicitly

A subcategory of $\mathbf{C} \subset \mathbf{Set}$ is one in which:
- $\mathbf{C}_0 \subset \mathbf{Set}_0$,
- for any $A, B \in \mathbf{C}_0$ we have $\mathrm{Hom}_{\mathbf{C}}(A,B) \subset \mathrm{Hom}_{\mathbf{Set}}(A,B)$,
- the operations $\circ_{\mathbf{C}}$ and $\mathrm{id}_{\mathbf{C}}$ and "inherited" from **Set**.

For example, let $\mathbf{C}$ be:



(The identity maps are not shown).

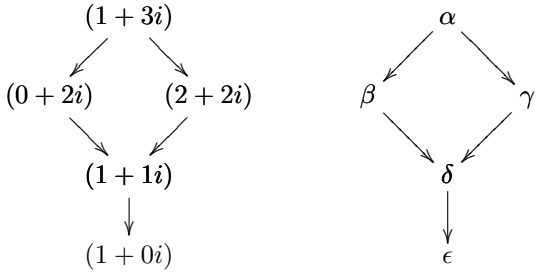
In this example $\mathbf{C}_0 = \{\, \{2,3\}, \{4,5,6\}, \{7,8\} \,\}$,
$\mathrm{Hom}_{\mathbf{C}}(\{2,3\},\{4,5,6\}) = \{\, \begin{smallmatrix} 2\mapsto 4 \\ 3\mapsto 5 \end{smallmatrix},\ \begin{smallmatrix} 2\mapsto 5 \\ 3\mapsto 4 \end{smallmatrix} \,\}$,
$\mathrm{Hom}_{\mathbf{C}}(\{2,3\},\{2,3\}) = \{\, \begin{smallmatrix} 2\mapsto 2 \\ 3\mapsto 3 \end{smallmatrix} \,\}$,
etc.

13. Well-positioned subsets of $\mathbb{N}^2$

By abuse of language, let's write
'$\mathbb{N}^2$' for $\mathbb{N} + i\mathbb{N}$ and
'$\mathbb{Z}^2$' for $\mathbb{Z} + i\mathbb{Z}$.

Example:
$\alpha = 1 + 3i$
$\beta = 0 + 2i$
$\gamma = 2 + 2i$
$\delta = 1 + 1i$
$\epsilon = 1 + 0i$

Let $K = \{ \alpha, \beta, \gamma, \delta, \epsilon \} \subset \mathbb{N}^2$
(mnemonic: $\underline{K}$ite).



We will say that a subset $D \subset \mathbb{Z}^2$ is *well-positioned*
when $\inf(\mathfrak{Re}(D)) = 0$ and $\inf(\mathfrak{Im}(D)) = 0$.

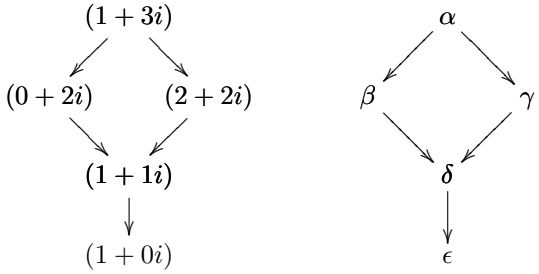
$$\begin{array}{rcl} \inf(\mathfrak{Re}(K)) & = & \inf(\mathfrak{Re}(\{\alpha, \beta, \gamma, \delta, \epsilon\})) \\ & = & \inf(\{1, 0, 2, 1, 1\}) \\ & = & 0 \\ \inf(\mathfrak{Im}(K)) & = & \inf(\mathfrak{Im}(\{\alpha, \beta, \gamma, \delta, \epsilon\})) \\ & = & \inf(\{3, 2, 2, 1, 0\}) \\ & = & 0 \end{array}$$

K is well-positioned.

14. Well-positioned subsets of $\mathbb{N}^2$ (2)

Def: an $\mathbb{N}^2$-*set* is a finite, well-positioned subset of $\mathbb{N}^2$.
We will denote $\mathbb{N}^2$-sets (with few elements) by bullet diagrams:
$K =$  , $V =$.

Def: to list the elements of an $\mathbb{N}^2$-set *in reading order*
is to list them from top to bottom, and from left to right.
Example: $K = \{\, \alpha, \beta, \gamma, \delta, \epsilon \,\}$.



We are interested in regarding $\mathbb{N}^2$-sets
as (the vertices of) directed (acyclical) graphs.

Def: an arrow $z \to w$ between points of $\mathbb{N}^2$ is a *black pawn's move*
when it moves at most one unit horizontally and exactly one unit down;
black pawn's moves are short arrows of shape $\swarrow$, $\downarrow$ or $\searrow$.

The set of black pawn's moves <u>on a subset</u> $D \subset \mathbb{N}^2$, $\mathsf{BPM}(D)$,
is defined in the obvious way:

$$\mathsf{BPM}(D) = \{\, (z, w) \in D^2 \mid z \to w \text{ is a black pawn's move} \,\}.$$

Example:
$\mathsf{BPM}(K) = \{\alpha \to \beta,\ \alpha \to \gamma,\ \beta \to \delta,\ \gamma \to \delta,\ \delta \to \epsilon\}$.

15. $\mathsf{BPM}(K)$ is not a protocategory

...because $\mathsf{BPM}(K)$ is just a set of arrows,
not a tuple of the form $\{K_0, \mathrm{Hom}, \circ, \mathrm{id}; \mathsf{assoc}, \mathsf{idLR}\}$...
What happens when we try to fix this?

Let's try:
- $K_0 = \{\alpha, \beta, \gamma, \delta, \epsilon\}$
- $\mathrm{Hom}_K(\alpha, \beta) = \{(\alpha, \beta)\}$, $\mathrm{Hom}_K(\alpha, \delta) = \{\ \}$, ...

16. $\mathsf{BPM}^*(K)$ is a protocategory

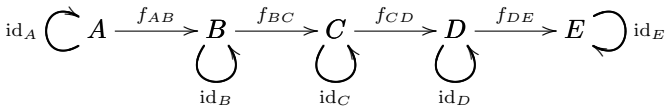
17. BPM*(K) is a category

18. Two categories generated by graphs

19. Commutative diagrams

20. Coherence

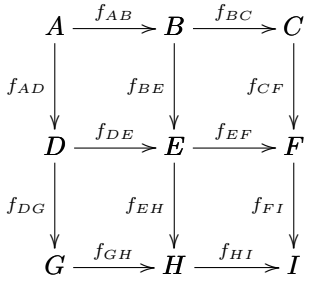
In a linear diagram like



all ways to define an arrow that "deserves the name f_{AE}"
give the same result.

[EXPLAIN WD]

In a planar diagram like the one below,
in which each cell commutes, i.e., wd$[f_{AE}]$, wd$[f_{BF}]$, wd$[f_{DH}]$, wd$[f_{EI}]$,
all ways to define an arrow that "deserves the name f_{AI}"
give the same result.



[EXERCISES]

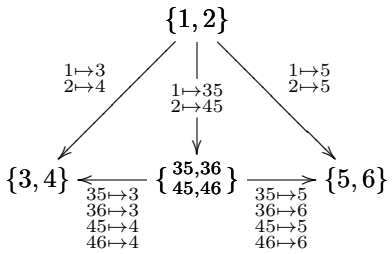
21. A terminal object in Set

Standard definition:
A *terminal object* in $\mathbf{C}$ is an object $T \in \mathbf{C}$
such that for each object $A \in \mathbf{C}$ there is exactly one map
(called t_A) from A to T.

Equivalent definition:
A *terminal* in $\mathbf{C}$ is a triple $(T, t; \mathsf{uniq})$ where:
- $T \in \mathbf{C}$,
- for each $A \in \mathbf{C}$ we have that $t(A)$ is a map from A to T,
- uniq is a proof for $\forall A. \forall f_{AT}. f_{AT} = t(A)$.

Fact 1: $\{200\}$ is a terminal object in **Set**.
EXERCISE: find the unique map from $\{2, 3, 4\}$ to $\{200\}$.
EXERCISE:
T, t, uniq
$\{\, 8 \,\}$ is a terminal in **Set**.

22. Products in Set



23. Terminals, categorically

24. Products, categorically

25. A category without products

26. Terminals are isomorphic

27. Products are isomorphic

28. A category without terminals and products

29. Categories with terminals and products

30. A functor in Set: Ax

31. A natural transformation in Set

32. Exponentials in Set

33. Exponentials, categorically

34. Adjunctions

35. Adjoints are isomorphic