

**Downcasing types**

(Introductory notes)

Eduardo Ochs

[eduardoochs@gmail.com](mailto:eduardoochs@gmail.com)<http://angg.twu.net/><http://angg.twu.net/math-b.html><http://angg.twu.net/LATEX/2007dnc-sets.{tex,dvi,pdf}>

Written: 2007.

L<sup>A</sup>T<sub>E</sub>Xing and minor additions and revisions: may 2008.

Current version: 2008may20 (see the footer)

Index of the slides:

Main idea: lifting archetypal proofs .....	2
Lifting set-like notation .....	3
Starting point: the Curry-Howard Isomorphism .....	4
Main ideas .....	5
Reconstructing the types and the dictionary from trees .....	6
Contexts and discharges .....	7
Contexts and discharges (2) .....	8
Expanding trees .....	9
Theorems for Free .....	10
Ambiguity plays on our side (usually) .....	11
Informal coherence .....	12
Informal coherence: examples .....	13
Informal coherence: examples (2) .....	14
Downcasing functors .....	15
Products via natural transformations .....	16
Downcasing Natural Transformations .....	17
Translating bigger trees .....	18
Downcasing categorical diagrams (in general) .....	19
Why we want to allow pseudopoints .....	20
Maps between subobjects: the general case .....	21
Quantifiers, categorically .....	22
More notes about archetypes (1) .....	23
More notes about archetypes (2) .....	24
More notes about archetypes (3) .....	25

## Main idea: lifting archetypal proofs

Take a proof of some general statement.  
 Now specialize it to a particular case.  
 ‘Specializing’ is somehow like doing a  
 projection - some distinctions collapse,  
 some details are lost -

Here we are concerned with the opposite  
 of specializing proofs.  
 We start with an ‘archetypal proof’ -  
 a proof of a certain very particular case,  
 done in a certain language - and then we  
 /change our dictionary a bit/ - and some  
 terms change meaning; and the same proof  
 becomes a proof of the general case -

We call this ‘lifting’. Archetypal proofs  
 /lift/ to more general proofs, and when this  
 happens we get general proofs done in a  
 language inherited from the archetypal cases.

My favorite examples of such liftings of  
 proofs are for proofs in Categorical  
 Semantics.

The first significant example will be this  
 one (we call it ‘ $CCC \leftrightarrow \lambda 1$ ’):

The Cartesian-Closed Categories (CCCs)  
 are exactly the categories where we can  
 interpret the simply-typed  $\lambda$ -calculus  
 (‘System  $\lambda 1$ ’).

After ‘ $CCC \leftrightarrow \lambda 1$ ’ we will see ‘ $Hyp \leftrightarrow LPCE$ ’:

The hyperdoctrines are exactly the  
 categories in which we can interpret  
 (a certain system of intuitionistic, typed)  
 first-order logic (‘LPCE’).

## Lifting set-like notation

Our approach:

'Set' is our archetypal CCC.  
 Sets are also our archetypal model for  $\lambda 1$ .  
 Most of the proof - even most of the details  
 of the statement of the theorem! - will appear  
 more or less naturally from taking the  
 diagrams below, done in the language of  
 'downcased types' (DNC) and translating  
 them in several ways.

$$\begin{array}{c}
 \text{--| a |--} \\
 / \quad | \quad \backslash \\
 v \quad v \quad v \\
 b \text{ <----| b,c |----> c}
 \end{array}
 \quad
 \begin{array}{c}
 a \\
 - \\
 | \\
 v \\
 *
 \end{array}
 \quad
 \begin{array}{c}
 a, b \text{ <==== a} \\
 - \\
 | \text{ <--> |} \\
 v \\
 c \text{ ===> b|->c}
 \end{array}$$

$$\begin{array}{c}
 \text{--| P |--} \\
 / \quad | \quad \backslash \\
 v \quad v \quad v \\
 Q \text{ <----| Q\&R |----> R}
 \end{array}
 \quad
 \begin{array}{c}
 P \\
 - \\
 | \\
 v \\
 \top
 \end{array}
 \quad
 \begin{array}{c}
 P \&Q \text{ <==== P} \\
 - \\
 | \text{ <--> |} \\
 v \\
 R \text{ ===> Q\text{-}R}
 \end{array}$$

Later, Set (or, more precisely, the fibration  $\text{Set}^{\text{>-}}$ ),  
 we will also be our archetypal hyperdoctrine.

## Starting point: the Curry-Howard Isomorphism

There is a correspondence between  
Natural Deduction trees and  
terms of the simply-typed  $\lambda$ -calculus ( $\lambda 1$ ).

Example:

	P&Q			d:A×B	
	---			-----	
P&Q	Q	Q⊃R	d:A×B	π'd:B	f:B→C
-----	-----	-----	-----	-----	-----
P	R		πd:A	f(π'd):C	
-----	-----		-----	-----	
	P&R			<πd, f(π'd)>:A×C	

Sometimes the usual notation for type theory/  
 $\lambda$ -calculus feels too verbose.

The DNC notation started as an informal reasoning tool -  
I used it in my private notes as a system of abbreviations  
for  $\lambda$ -calculus.

DNC grew with time and started to look more formal...

It got special arrow symbols (a.k.a. ‘connectives’ - ‘=>’ and  
‘-.>’) to represent functors and natural transformations.

In 2001 I found that ‘=>’ and ‘-.>’ had introduction rules  
besides the obvious elimination rules (think in introd and  
elim of ‘&’, ‘⊃’, ‘\or’, etc in Natural Deduction) -  
so DNC could be the basis for a system of natural deduction  
for categories.

DNC never became formal enough to be a ‘system’, and nowadays it  
lives a more unpretentious existence as an auxiliary language for  
type theory and category theory (esp. categorical semantics).

I changed the meaning for ‘DNC’ -  
from a sytem for ‘Natural Deduction for Categories’ to  
a notation of ‘DownCased types’.

## Main ideas

Long names for variables

-----  
 In  $\{(n,r) \in \mathbb{N} \times \mathbb{R} \mid r^2 = n\}$   
 the  $(n,r)$  is the name of a variable -  
 $\mathbb{N} \times \mathbb{R}$  is the space of pairs made of a  
 natural number and a real.

A dictionary

-----  
 Define  $n := \pi(n,r)$   
        $r := \pi'(n,r)$   
        $(r,n) := \langle r,n \rangle$

The syntactical distinction between variables and terms

-----  
 We've sacrificed it.  
 Also, without the dictionary there is no way to tell from  
 $(n,r)$ ,  $n$ ,  $r$ ,  $(r,n)$  which of them are more primitive than  
 the others.

Downcasing

-----  
 The default name for a variable ranging over  $A$  is 'a'.  
 The default (long!) name for a variable ranging over  $A \times B$  is '(a,b)'.

Well-formed formulas, expressions, etc

-----  
 We don't have these notions in DNC.  
 Anything can be a DNC 'name'. 8-(

Spaces of functions

-----  
 In  $f:A \rightarrow B$  we will read ' $A \rightarrow B$ ' as  $B^A$   
 and the ':' as the ':' of Type Theory.  
 (For mathematicians: ':' is roughly like ' $\in$ ', but ' $f:A \rightarrow B$ '  
 means also "'f' is of type ' $A \rightarrow B$ '", and each term has  
 a single type. We are not interesting in '\subset', '\subteq', etc).

Notation for functions

-----  
 An ' $a \mapsto b$ ' is an element of  $A \rightarrow B$ .  
 An ' $a \mapsto b$ ' is a function that takes each 'a' to a 'b'.

### Reconstructing the types and the dictionary from trees

	d:A×B	
	-----	
d:A×B	π'd:B	f:B->C
-----	-----	
πd:A	f(π'd):C	
-----		
	<πd,f(π'd)>:A×C	
	a,b	
	---	
a,b	b	b ->c
---	-----	
a	c	a := π (a,b)
-----		b := π' (a,b)
		c := (b ->c)(b)
		a,c := <a,c>
	a,c	

Now add ‘a,b := d’ and ‘b|->c := f’ and we get the λ-calculus terms back from the DNC tree. Note that we obtain the types back by uppercasing the DNC ‘terms’ -

- ‘|->’ is uppercased to ‘->’,
- ‘,’ to ‘×’, etc.

## Contexts and discharges

Each subtree of a Natural Deduction tree/derivation/proof corresponds to a smaller proof contained in a bigger one. For each node of a tree let's look at the tree above the node, and make its hypotheses explicit.

$P \& Q$	$P \& Q \mid -P \& Q$
---	-----
$P \& Q$ $Q$ $Q \supset R$	$P \& Q \mid -P \& Q$ $P \& Q \mid -Q$ $Q \supset R \mid -Q \supset R$
-----	-----
$P$ $R$	$P \& Q \mid -P$ $Q \supset R, P \& Q \mid -R$
-----	-----
$P \& R$	$Q \supset R, P \& Q \mid -P \& R$

Now we can understand the ' $\supset$ -introduction' rule; it involves a discharge.

$[P \& Q]^1$	$P \& Q \mid -P \& Q$
-----	-----
$[P \& Q]^1$ $Q$ $Q \supset R$	$P \& Q \mid -P \& Q$ $P \& Q \mid -Q$ $Q \supset R \mid -Q \supset R$
-----	-----
$P$ $R$	$P \& Q \mid -P$ $Q \supset R, P \& Q \mid -R$
-----	-----
$P \& R$	$Q \supset R, P \& Q \mid -P \& R$
-----1	-----
$P \& Q \supset P \& R$	$Q \supset R \mid -P \& Q \supset P \& R$

Below the bar marked with '1' the hypotheses marked with ' $[ \cdot ]^1$ ' should no longer be in the list of hypotheses - they've been 'discharged' (into the conclusion).

## Contexts and discharges (2)

Let's look at the functional side of this (Curry-Howard):

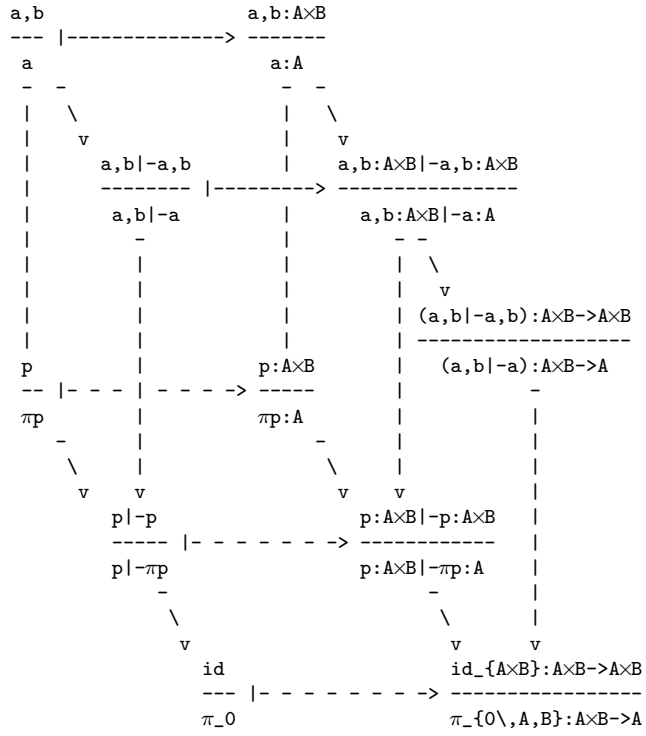
$$\begin{array}{c}
 \frac{[a,b]^1 \quad b \quad b \rightarrow c}{a \quad c} \quad \frac{(a,b) | -(a,b) \quad (a,b) | -b \quad (b \rightarrow c) | -(b \rightarrow c)}{(a,b) | -a \quad (b \rightarrow c), (a,b) | -c} \\
 \frac{a, c}{a, b \rightarrow a, c} \quad \frac{(b \rightarrow c), (a,b) | -(a,c)}{(b \rightarrow c) | -(a, b \rightarrow a, c)} \\
 \\
 \frac{[p]^1 \quad \pi'p \quad f}{\pi p \quad f(\pi'p)} \quad \frac{p | -p \quad p | -\pi'p \quad f | -f}{p | -\pi p \quad f, p | -f(\pi'p)} \\
 \frac{\langle \pi p, f(\pi'p) \rangle}{\lambda p. \langle \pi p, f(\pi'p) \rangle} \quad \frac{f, p | -\langle \pi p, f(\pi'p) \rangle}{f | -\lambda p. \langle \pi p, f(\pi'p) \rangle}
 \end{array}$$

The ‘ $\rightarrow$ -introduction’ rule (last bar) corresponds to introducing a ‘ $\lambda$ ’; one variable ceases to be free, and is removed from the list of hypotheses.

One way of reading ‘ $f, p | -\langle \pi p, f(\pi'p) \rangle$ ’ -  
 or ‘ $f: B \rightarrow C, p: A \times B | -\langle \pi p, f(\pi'p) \rangle: A \times C$ ’ - is:  
 if we know the value of  $f$  and  $p$  (points of  $B \rightarrow C$  and  $A \times B$ )  
 we know the value of  $\langle \pi p, f(\pi'p) \rangle$  (a point of  $A \times C$ ).



## Expanding trees



## Theorems for Free

From the ‘‘Theorems for Free’’ paper (Wadler 1989):

- \* All closed terms of type  $\forall A.\forall B.(A \times B \rightarrow B \times A)$  obey a certain theorem
- \* Corollary: all terms of type  $\forall A.\forall B.(A \times B \rightarrow B \times A)$  are the flip function.

Something similar (but much looser) happens in DNC:

‘ $a, b \mid \rightarrow b, a$ ’ has a ‘‘natural definition’’ -  
 apply Curry-Howard, prove  $P \& Q \dashv\vdash Q \& P$ , translate the proof to  $\lambda$ -calculus in DNC notation -

$[P \& Q] \sim 1$	$[P \& Q] \sim 1$	$[a, b] \sim 1$	$[a, b] \sim 1$	$[p] \sim 1$	$[p] \sim 1$
Q	P	b	a	$\pi'p$	$\pi p$
Q&P		b, a		$\langle \pi'p, \pi p \rangle$	
-----1		-----1		-----1	
$P \& Q \dashv\vdash Q \& P$		$a, b \mid \rightarrow b, a$		$\lambda p. \langle \pi'p, \pi p \rangle$	

and an argument using normalization of ND proofs can be used to show that all other ND proofs (=  $\lambda$ -terms) are equivalent to that one.

In DNC we won't be interested in the ‘‘uniqueness’’ part very often, though.

### Ambiguity plays on our side (usually)

a,b ---	a,b ---	p --	p ---	a := $\pi$ (a,b)
a	b	$\pi p$	$\pi'p$	b := $\pi'$ (a,b)
-----				a,b := <a,b>
a,b				< $\pi p, \pi'p$ >

Apparently the DNC tree on the left, above introduces a ‘‘secondary definition’’ for a,b and a circularity in the dictionary, and such things have to be avoided at all costs -

But we can be careful and interpret each tree as a term, and in

a,b ---	a,b ---	p --	p ---	
a	b	$\pi p$	$\pi'p$	
-----				
a,b = a,b				< $\pi p, \pi'p$ > = p

what happens is that we have two different ‘‘natural constructions’’ for a,b from a,b, and we’re saying that the two ‘‘must give the same result’’, i.e.,  $\langle \pi p, \pi'p \rangle = p$ ...

In many contexts we will have two different natural constructions for a DNC ‘‘term’’ with a certain name, and we will want to say that the two constructions give the same result (sometimes this will be an axiom, sometimes a theorem, sometimes a hypothesis) -

We will sometimes use the expression ‘‘is well defined’’ (notation: ‘‘wd[.]’’) in a funny sense: ‘‘wd[a,b]’’ will mean ‘‘the two obvious natural constructions for a,b give the same result’’.

## Informal coherence

A /coherence theorem/ says that there is exactly one entity of a given type...

For example: that ‘‘there is exactly one map  $A \times B \rightarrow B \times A$ ’’ - the flip function.

Coherence would allow us to say ‘‘\_the\_map  $a, b \rightarrow a, b$ ’’ instead of ‘‘\_a\_map  $a, b \rightarrow a, b$ ’’.

We have something weaker in DNC: a dictionary, and a vague notion of assigning ‘‘natural meanings’’ to names.

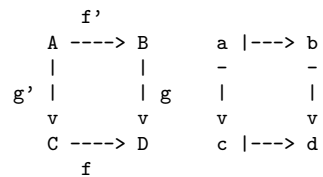
Each ‘‘natural meaning’’ comes from a ‘‘natural construction’’.

Instead of proving ‘‘real’’ coherence - e.g., ‘‘all natural constructions for  $a, b \rightarrow b, a$  give the same result’’ - we will use something MUCH weaker:

‘‘all the ‘obvious’ natural constructions for  $a, b \rightarrow b, a$  (in a given context - given some base objects, operations, and diagrams) yield the same result.’’

### Informal coherence: examples

If we have set meanings for the four arrows in the diagram below,



then we have two different ‘‘natural constructions’’ for  $a \dashrightarrow d$ , each one giving a (possibly different) ‘‘natural meaning’’ for  $a \dashrightarrow d$ ...

‘‘ $\text{wd}[\alpha]$ ’’ (pronounced:  
‘‘ $\alpha$  is well-defined’’)

means:

‘‘all the obvious natural constructions for  $\alpha$  yield the same result’’

In the diagram above  
 ‘‘ $\text{wd}[a \dashrightarrow d]$ ’’ means  $f'; g=g'; f$   
 (i.e.,  $a \dashrightarrow b \dashrightarrow d = a \dashrightarrow c \dashrightarrow d$ )  
 i.e.: ‘‘the square commutes’’.

Giving meaning to a ‘‘wd’’ expression involves deciding which are the ‘‘obvious natural constructions’’.

## Informal coherence: examples (2)

Each sentence ‘ $\text{wd}[\backslash\alpha]$ ’ is a proposition whose precise meaning is given as an entry in the dictionary.

Another example: in

$$\begin{array}{ccccc} a & | \dashrightarrow & b & | \dashrightarrow & c \\ - & & - & & - \\ | & & | & & | \\ v & & v & & v \\ d & | \dashrightarrow & e & | \dashrightarrow & f \\ - & & - & & - \\ | & & | & & | \\ v & & v & & v \\ g & | \dashrightarrow & h & | \dashrightarrow & i \end{array}$$

we have:

$$\begin{array}{l} \text{wd}[a \rightarrow e] \ \& \ \text{wd}[b \rightarrow f] \ \& \\ \text{wd}[d \rightarrow h] \ \& \ \text{wd}[e \rightarrow i] \ \Rightarrow \ \text{wd}[a \rightarrow i]. \end{array}$$

Note that:

$$\begin{array}{l} \text{wd}[a \rightarrow i] \ := \ (a \rightarrow b \rightarrow c \rightarrow f \rightarrow i) = \\ \quad a \rightarrow b \rightarrow e \rightarrow f \rightarrow i = \\ \quad a \rightarrow b \rightarrow e \rightarrow h \rightarrow i = \\ \quad a \rightarrow d \rightarrow e \rightarrow f \rightarrow i = \\ \quad a \rightarrow d \rightarrow e \rightarrow h \rightarrow i = \\ \quad a \rightarrow d \rightarrow g \rightarrow h \rightarrow i). \end{array}$$

Each of the other ‘ $\text{wd}[\cdot]$ ’s is an equality between only two composites (not six).

## Downcasing functors

A functor  $F:\text{catC}\rightarrow\text{catD}$  is composed of two operations:

$F_0$  - its action on objects, and

$F_1$  - its action on morphisms.

Fix a set  $B$ , and look at these diagrams, for the functors

$(\times B):\text{Set}\rightarrow\text{Set}$  and  $(B\rightarrow):\text{Set}\rightarrow\text{Set}$ :

$$\begin{array}{ccc}
 A \times B & \longleftarrow & A & & a, b & \longleftarrow & a \\
 \hline
 f \times B & \longleftarrow & f & & & \longleftarrow & \\
 \hline
 A' \times B & \longleftarrow & A' & & a', b & \longleftarrow & a'
 \end{array}$$
  

$$\begin{array}{ccc}
 C & \longleftarrow & B \rightarrow C & & c & \longleftarrow & b \rightarrow c \\
 \hline
 g & \longleftarrow & B \rightarrow g & & & \longleftarrow & \\
 \hline
 C' & \longleftarrow & B \rightarrow C' & & c' & \longleftarrow & b \rightarrow c'
 \end{array}$$

Functors act on two levels, so we will use a double arrow - ' $\Rightarrow$ ' - to denote them. Their names in DNC will come from their action on objects:

$$\begin{array}{ll}
 A \rightarrow A \times B & a \Rightarrow a, b \\
 C \rightarrow B \rightarrow C & c \Rightarrow b \rightarrow c
 \end{array}$$

Their 'syntactical actions' on morphisms can be derived from their names:  $a \Rightarrow a, b$  adds a ' $, b$ ' to the name of each object, so it takes  $a \rightarrow a'$  to  $a, b \rightarrow a', b$ ; same for  $c \Rightarrow b \rightarrow c$ .

To find the 'real action' of  $(\times B) = (a \Rightarrow a, b)$  on morphisms, look for a natural construction of  $a, b \rightarrow a', b$  from  $a \rightarrow a'$ ;  $f \times B = a, b \rightarrow a', b = \lambda p. \langle f(\pi p), \pi' p \rangle$ .

Note that a functor  $a \Rightarrow a, b$  does not take 'a's into 'a, b's, like an arrow  $a \rightarrow a, b$  would do; instead, it takes the whole space of 'a's,  $E[a]=A$ , into the space of 'a, b's,  $E[a, b]=A \times B$  -  $(\times B)_0 = \lambda A:\text{Sets}. A \times B$ .

### Products via natural transformations

Any diagram  $b \leftarrow p \rightarrow c$  induces an operation that takes any object  $A$  to a map of sets

$$(A \rightarrow P) \rightarrow (A \rightarrow B) \times (A \rightarrow C)$$

$$\begin{array}{ccc}
 / a \backslash & & / \quad a \quad \backslash \\
 | - | & \rightarrow & | \quad - \quad | \\
 | | | & & | \quad / \quad \backslash \quad | \\
 | v | & & | \quad v \quad \quad v \quad | \\
 \backslash p / & & \backslash b \quad \quad \quad c /
 \end{array}$$



## Downcasing Natural Transformations

Between two contravariant functors,

$$\begin{aligned} \text{Hom}(-, P) : \text{Set}^{\text{op}} &\rightarrow \text{Set} \\ A &\mapsto A \rightarrow P \\ a^{\text{op}} &\Rightarrow a \mapsto p \end{aligned}$$

$$\begin{aligned} \text{Hom}(-, B) \times \text{Hom}(-, C) : \text{Set}^{\text{op}} &\rightarrow \text{Set} \\ A &\mapsto (A \rightarrow B) \times (A \rightarrow C) \\ a^{\text{op}} &\Rightarrow (a \mapsto b), (a \mapsto c) \end{aligned}$$

The natural transformation

$$T : \text{Hom}(-, P) \rightarrow \text{Hom}(-, B) \times \text{Hom}(-, C)$$

takes each object  $A$  of  $\text{Set}$  (or: an object of  $\text{Set}^{\text{op}}$ ) to a function:

$$TA : (A \rightarrow P) \rightarrow ((A \rightarrow B) \times (A \rightarrow C))$$

We will represent this natural transformation  $T$  in DNC by just downcasing the triangle:

$$\begin{array}{c} A \\ \text{---} \\ / \quad | \quad \backslash \\ v \quad v \quad v \\ (A \rightarrow P) \rightarrow ((A \rightarrow B) \times (A \rightarrow C)) \end{array}$$

### Translating bigger trees

$$\begin{array}{ccc}
 \begin{array}{c}
 \frac{a,b}{\text{---}} \\
 \frac{a,b \quad a \quad a|-b| \rightarrow c}{\text{---}} \\
 \frac{b \quad b| \rightarrow c}{\text{---}} \\
 c
 \end{array}
 & \xrightarrow{\quad} &
 \begin{array}{c}
 \frac{a,b|-a,b}{\text{---}} \\
 \frac{a,b|-a,b \quad a,b|-a \quad a|-b| \rightarrow c}{\text{---}} \\
 \frac{a,b|-b \quad a,b|-b| \rightarrow c}{\text{---}} \\
 a,b|-c
 \end{array} \\
 \begin{array}{c}
 - \\
 | \\
 v
 \end{array}
 & &
 \begin{array}{c}
 - \\
 | \\
 v
 \end{array} \\
 \\
 \begin{array}{c}
 \frac{p \quad a|-f(a)}{\text{---}} \\
 \frac{p \quad \pi p \quad \pi p|-f(\pi p)}{\text{---}} \\
 \frac{\pi' p \quad f(\pi p)}{\text{---}} \\
 f(\pi p) (\pi' p)
 \end{array}
 & \xrightarrow{\quad} &
 \begin{array}{c}
 \frac{p|-p \quad a|-f(a)}{\text{---}} \\
 \frac{p|-p \quad p|-p \quad \pi p|-f(\pi p)}{\text{---}} \\
 \frac{p|-p \quad p|-f(\pi p)}{\text{---}} \\
 p|-f(\pi p) (\pi' p)
 \end{array} \\
 \\
 \begin{array}{c}
 \text{id} \\
 \text{---} \\
 \text{id} \quad \pi \quad f \\
 \text{---} \quad \text{---} \\
 \pi' \quad \pi;f \\
 \text{---} \\
 \langle \pi;f,\pi' \rangle; \text{ev} = f^{\wedge} v
 \end{array}
 \end{array}$$

## Downcasing categorical diagrams (in general)

One day I realized that the same notation that I was using for sets did also work - with no changes in the syntax, only in the semantics - for categories.

Here's a comparison; in the second situation, at the right,  $\text{\catC}$  is an arbitrary category.

A is a set	A is an object of $\text{\catC}$ , $A \in  \text{\catC} $
B is a set	B is an object of $\text{\catC}$ , $B \in  \text{\catC} $
A $\dashrightarrow$ B (a function)	A $\dashrightarrow$ B (a morphism)
a $\mapsto$ b (a function)	a $\mapsto$ b (a morphism)
A is the space of 'a's	Now:
B is the space of 'b's	"an 'a'" is an abuse of language
A = E[a]	"'a 'b'" is an abuse of language
B = E[b]	'a $\mapsto$ b' has semantics - it's a morphism -
	but 'a' and 'b' have no semantics
An 'a' is a point of A,	'a's and 'b's are not points in this case,
a 'b' is a point of B,	just 'pseudopoints', and pseudopoints
$a \in A$ , $b \in B$ .	don't need to exist - they may be just
	syntactical devices
	To enforce the distinction we say
	"object of" instead of "space of"
	and we write $O[\cdot]$ instead of $E[\cdot]$
	A is the objects of 'a's
	B is the objects of 'b's
	A = $O[a]$
	B = $O[b]$

Note: it's almost impossible to understand this is the general case without examples...

## Why we want to allow pseudopoints

For example, for subobjects.  
Implications can be seen as inclusions.

$$\forall a \in A. P(a) \sqsubseteq Q(a) \\ \{a \in A \mid P(a)\} \subseteq \{a \in A \mid Q(a)\}$$

Categorically,

$$\begin{array}{ccc} \{a \mid P(a)\} & \dashrightarrow & \{a \mid Q(a)\} \\ \downarrow & & \downarrow \\ \{a \mid P(a)\} \setminus & & \{a \mid Q(a)\} \setminus \\ \downarrow & \dashrightarrow & \downarrow \\ \{a \mid P(a)\} & & \{a \mid Q(a)\} \\ \downarrow & & \downarrow \\ A & \dashrightarrow & A \end{array}$$

This is a morphism between two objects of  $\text{Set}^{\rightarrow}$ , where  $\text{Set}^{\rightarrow}$  is the category of monic arrows between objects of  $\text{Set}$ .  $\text{Set}^{\rightarrow}$  is also called  $\text{Sub}(\text{Set})$  - the category of subobjects of  $\text{Set}$ .

Notational trick:

$$\{a \mid P(a)\} \dashrightarrow \{a \mid Q(a)\} := \begin{array}{ccc} \{a \mid P(a)\} \setminus & & \{a \mid Q(a)\} \setminus \\ \downarrow & \dashrightarrow & \downarrow \\ \{a \mid P(a)\} & & \{a \mid Q(a)\} \\ \downarrow & & \downarrow \\ A & & A \end{array}$$

The ' $\mid$ ' in  $\{a \mid P(a)\}$  is called the "such that" bar;  
we're doubling it to represent subobjects.  
We downcase that into:

$$a \mid P(a) \mid \dashrightarrow a \mid Q(a)$$

this morphism - in  $\text{Set}^{\rightarrow}$  is not just a mapping of points...  
in fact, it's two morphisms in  $\text{Set}$  -  $a \mid P(a) \mid \dashrightarrow a \mid Q(a)$  and  $a \mid \dashrightarrow a$  -  
plus the information that the square below commutes:

$$\begin{array}{ccc} a \mid P(a) \mid & \dashrightarrow & a \mid Q(a) \\ / & & / \\ \mid & & \mid \\ \downarrow & & \downarrow \\ a & \dashrightarrow & a \end{array}$$

### Maps between subobjects: the general case

More generally,  
 $\{a|P(a)\} \rightarrow \{b|Q(b)\}$  is:

$$\begin{array}{ccc}
 & & b:=b(a) \\
 \{a|P(a)\} \backslash & & / \\
 | \quad v \quad | & \longrightarrow & | \quad v \quad | \\
 | \quad | \quad | & & | \quad | \quad | \\
 | \quad v \quad | & & | \quad v \quad | \\
 \backslash \quad A \quad / & & \backslash \quad B \quad /
 \end{array}
 \quad
 \begin{array}{ccc}
 a|P(a) & | \xrightarrow{\quad} & b|Q(b) \\
 / & & / \\
 | & & | \\
 v & \quad b:=b(a) & v \\
 a & | \xrightarrow{\quad} & b
 \end{array}$$

$a|P(a) \mapsto b|Q(b)$  means  $\forall a. P(a) \Rightarrow Q(b(a))$ .

Note the use of a bit of ‘‘physicist’s notation’’ here - in the presence of a function  $f = a \mapsto b$  the value of  $b$  can be seen as a function of  $a$ , and a physicist would write  $b=f(a)$  or  $b=b(a)$ ... we will prefer  $b=b(a)$  to avoid using new letters, and the dictionary will translate such expressions (atrocities!) into something mathematically sound.

We will usually write pseudopoints like  $a|P(a)$  as ‘‘ $P(a)$  over  $a$ ’’ (in parentheses), and we will draw these things over their ‘‘codomain projections’’:

$$\begin{array}{ccc}
 \{a|P(a)\} \longrightarrow \{b|Q(b)\} & /P(a) \backslash & | \longrightarrow | \\
 - & \backslash a / & \backslash b / \\
 \text{cod } | & & | \\
 v & \quad f & v \\
 A \xrightarrow{\quad} B & & a \xrightarrow{\quad} b
 \end{array}$$

In some rare occasions, when space is at premium, we will write just  $P(a)$  instead of ‘‘ $P(a)$  over  $a$ ’’. The ‘‘codomain projection’’ functor arrows will usually not be drawn.

$$\begin{array}{ccc}
 \{a|P(a)\} \longrightarrow \{b|Q(b)\} & P(a) & | \longrightarrow | \\
 & f & \\
 A \xrightarrow{\quad} B & & b:=b(a) \\
 & & a \xrightarrow{\quad} b
 \end{array}$$

## Quantifiers, categorically

In  $\text{Set}^{\rightarrow}$  the functors that quantify over a variable are adjoints to ‘weakening functors’ that add a dummy variable to the domain:

$$\begin{array}{ccc}
 \{a,b \mid P(a,b)\} \dashrightarrow \{a \mid \exists b.P(a,b)\} & P(a,b) \iff \exists b.P(a,b) \\
 \downarrow & \downarrow \\
 \{a,b \mid Q(a)\} \dashrightarrow \{a \mid Q(a)\} & Q(a) \iff Q(a) \\
 \downarrow & \downarrow \\
 \{a,b \mid R(a,b)\} \dashrightarrow \{a \mid \forall b.R(a,b)\} & R(a,b) \iff \forall b.R(a,b) \\
 \downarrow & \downarrow \\
 A \times B \dashrightarrow A & a, b \dashrightarrow a
 \end{array}$$

These functors do not operate on the whole of  $\text{Set}^{\rightarrow}$  - they go from one ‘fiber’ of  $\text{Set}^{\rightarrow}$  to another.

The best way to formalize this is via /fibrations/;  $\text{Set}^{\rightarrow}$  is a fibration over  $\text{Set}$ .

A fibration is composed of:  
 an ‘entire category’  $\text{Set}^{\rightarrow}$   
 a ‘base category’  $\text{Set}$   
 a ‘projection functor’  $\text{Cod: } \text{Set}^{\rightarrow} \rightarrow \text{Set}$   
 and enough ‘cartesian liftings’. (?!?!?! - next slide)

In our diagrams we will not usually draw the projection functors. Instead we will just draw the objects of the entire category over their projections - and we will often omit the part of their names that can be recovered from the projections. In the diagram above we wrote just ‘ $Q(a)$ ’, not ‘ $a \mid Q(a)$ ’.

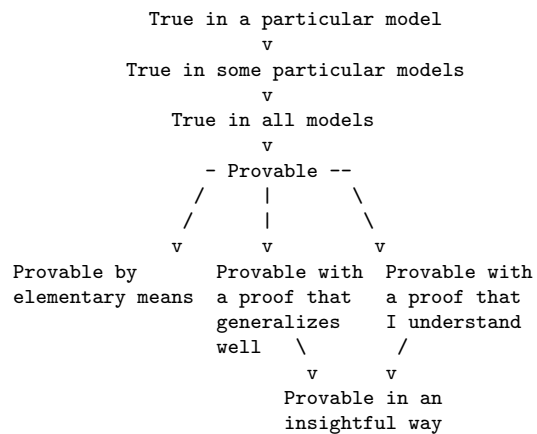
## More notes about archetypes (1)

There are different levels of mathematical truths - some ideas are "obvious", and I can use them implicitly or by just mentioning them briefly, and my interlocutor will accept them; other ideas - for example, theorems that have been published, but whose proofs are hard - I can only use with extreme care, and a proof that makes use of these hard theorems automatically gets marked with a small red or black flag -

A statement like "the composite  $(a,b) \rightarrow (b,a) \rightarrow (a,b)$  is the identity" takes very little "mental space", and is intuitively true, even though in different settings it may have different proofs, and some of these proofs - in  $\lambda$ -calculus, for example - may look too technical at first...

Where do these "intuitively true statements" live?  
What languages can we develop to work with them?

Here is a diagram showing some kinds of mathematical truths. The arrows go from "weaker truths" to "stronger truths".



## More notes about archetypes (2)

(Note:  $D$  stands for a DAG - usually finite, and in most examples having at most five vertices)

For me, this is the archetype behind the idea of "topos":

Every  $\text{Set}^D$  is a topos.

However, the resulting concept is much more general than that - and the "archetypal language" for a topos comes from a very particular case -  $\text{Set}^D = \text{Set}$ ,  $D =$  the one-point DAG.

This is the archetype of a "sheaf":

Every  $\text{Set}^{\mathcal{O}(D)^{\text{op}}}$  has a notion of ' $\text{sheaf}$ '. The resulting notion of ' $\text{sheaf}$ ' - based on Lawvere-Tierney topologies and modalities - admits several other kinds of sheaves: ' $\text{sheaves}$ ', ' $\text{sheaves}$ ', ' $\text{sheaves}$ ', and forcing.

The precise definition of what a topos is can be expressed in the language of the archetype; we can use the language of the archetype to express most (i.e., "enough") of the most important constructions and theorems; we can use it to help us build dictionaries between the standard, more precise notations; and proofs done in the archetype take relatively little mental space, and generalize easily. Also, the archetypal language for a topos may collapse (in the syntax!) some things that are different in some toposes, but it is "non-trivial enough" - it does not collapse too much.



### More notes about archetypes (3)

What do I expect from an archetype and its language?

"It should be expressive enough"

Most of the basic theory - constructions and theorems - can be done in the archetypal language, and then lifted to the general case; the general definition can be expressed using the archetypal language.

"It collapses enough"

The archetypal syntax identifies (in the "wd[.]" sense) many constructions that are expected to give the same results.

"It is non-trivial enough"

Even though the archetypal syntax may identify some constructions that are distinct in some models (see the example " $x,ab,cd \leftarrow |x,a,b,c,d$ " in monads) it does not collapse too much.

Also, it can't take up too much "mental space" -