Downcasing Types

Eduardo Ochs
PURO/UFF, LLaRC
eduardoochs@gmail.com
http://angg.twu.net/

I first presented this, in an incomplete form,
at the UniLog'2010 (Cascais, Portugal, april 2010)...
The current version (see the footer for the date)
has many more slides, and I am still adding more.

Index of the slides:

LLaRC
LLaRC =
Laboratório de Lógica e Representação do Conhecimento =
Laboratory of Logic and Representation of Knowledge
at PURO/UFF =
Pólo Universitário de Rio das Ostras/
Universidade Federal Fluminense

Rio das Ostras is city in the countryside of the
state of Rio de Janeiro, Brazil.
RO is ∼180Km away from Rio de Janeiro
and ∼150Km away from Niterói,
which is where the main campi of UFF are situated.

The LLaRC is a research laboratory that,
despite our efforts in the last two years, 8-(,
still doesn't have a physical room,
(the PURO has been facing severe space problems),
neither a decent internet connection
(we still have only about 3KB/s per machine at PURO),
and from nov/2009 to mar/2010 none of the printers
at PURO had toner...

We have decided that in order to give
some kind of official existence to our current research
we would temporarily lower the bar for what we would "publish",
and work-in-progress versions of seminar notes, like these,
could be put in the LLaRC home page.

(We do not get brownie points for these).

Outsiders
This is Category Theory (from here on: "CT")
done by an outsider, in a place where no one else
is doing research in CT, or even using CT in non-trivial ways...

If the ideas in these notes were perfect
AND crystal-clear AND had the best impact possible
then this is more or less what would happen:

• Local people close to me (non-CT-ists) would become able to read CT books and articles;

• CT-ists *(would...)* find the method described here trivial, and would start to circulate notes describing the archetypical models behind several constructions that I've been struggling to understand (sheafification, classifying toposes, *-autonomous categories, differential categories, Kan extensions, the Grothendieck construction on a fibration, schemes, parametric models, DTT on LCCCs, etc); I get hold of their notes and understand everything;

• Type theorists find out the right type systems for the syntactical world, the real world and the projection; they also find classes of type equations (for the "hints") that can be extracted from the diagrams and solved automatically;

• Proof-assistants people come up with very good ways to formalize CT proofs by doing the syntactical part first, then completing the details;

• Parametricity people grok immediately everything here, and we work together on the missing meta-theorems;

• Philolophical Logic people would help me to fit these ideas in a grander scheme of things: synthetical vs. analytical reasoning, external diagrams vs. internal diagrams, the roles of intuition, of archetypal models and generalization;

• Diagrammatic logic people find out how to enrich my categorical diagrams with some extra hints, so that diagrams like these, when stored in a computer with the those extra hints would automatically generate the derivation trees — and the code for proof-assistants.

Outline

- Introduction
- The Yoneda Lemma
- Monads
- Cartesian Closed Categories
- Hyperdoctrines
- Typing (proto-)fibrations

Curry-Howard

$$\dfrac{\dfrac{\dfrac{[P\land Q]^1}{P} \qquad \dfrac{\dfrac{[P\land Q]^1}{Q} \qquad Q\supset R}{R}}{P\land R}}{P\land Q\supset P\land R}\,1
\qquad
\dfrac{\dfrac{\dfrac{[p:A\times B]^1}{\pi p:A}\,\pi \qquad \dfrac{\dfrac{[p:A\times B]^1}{\pi' p:B}\,\pi' \qquad f:B\to C}{f(\pi' p):C}\,\mathrm{app}}{\langle \pi p, f(\pi' p)\rangle : A\times C}\,\langle,\rangle}{\lambda p{:}(A\times B).\langle \pi p, f(\pi' p)\rangle : A\times B\to A\times C}\,\lambda;1$$

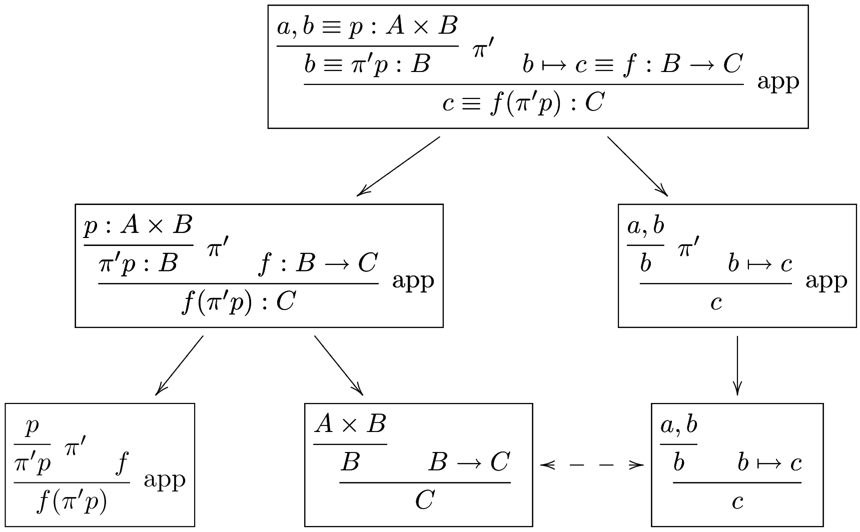
$$\dfrac{\dfrac{\dfrac{[a,b]^1}{a} \qquad \dfrac{\dfrac{[a,b]^1}{b} \qquad b\mapsto c}{c}}{a,c}}{a,b\mapsto a,c}\,1
\qquad
\dfrac{\dfrac{\dfrac{[p:A\times B]^1}{\pi p:A}\,\pi \qquad \dfrac{\dfrac{[p:A\times B]^1}{\pi' p:B}\,\pi' \qquad f:B\to C}{f(\pi' p):C}\,\mathrm{app}}{\langle \pi p, f(\pi' p)\rangle : A\times C}\,\langle,\rangle}{\lambda p{:}(A\times B).\langle \pi p, f(\pi' p)\rangle : A\times B\to A\times C}\,\lambda;1$$

The Curry-Howard isomorphism
(in its simplest form — there are several)
establishes a correspondence between
trees in (propositional) Natural Deduction and
trees in (simply-typed) λ-calculus.

λ-calculus is *the* language for expressing constructions.
The terms in the bottom of λ-calculus trees get bigger and bigger,
so people usually don't draw λ-calculus trees —
they work "algebraically", just with term at the bottom.
A λ-calculus tree can be reconstructed from its bottom term.
Humans like full λ-calculus trees, though,
because the trees explain the types of all subterms.

The tree at the bottom left above
describes the "intuitive meaning" behind
the λ-calculus tree.

Erasings as projections



The "downcased tree" at the bottom right above
is an "intuitive view" of the (λ-calculus) construction.
How do we formalize an "intuitive construction" like that?
Answer: *erasings* act like *projections*.
We add the missing information by *lifting*
and then we erase some things.

Here's how we read the "downcased tree" aloud.
"If we have a meaning for 'b' and a meaning for '$b \mapsto c$'
we have a natural meaning for 'c'"...
Linguistic tricks:

- long names: '$b \mapsto c$' is a *name*
- no syntactical distinction between variables and non-atomic terms
- '$b \mapsto c$' to '$B \to C$': the types can be recovered by *uppercasing*
- each bar is a *definition*
- indefinite articles: "if I have a 'b' and a '$b \mapsto c$' I have a 'c'"...

Generalization is a kind of lifting
To generalize is the opposite of to specialize,
and specialization is obviously a projection.

$$\boxed{\begin{aligned} 2^{n+1} - 2^n &= 2^{1+n} - 2^n \\ &= 2^1 \cdot 2^n - 2^n \\ &= 2 \cdot 2^n - 1 \cdot 2^n \\ &= (2-1) \cdot 2^n \\ &= 1 \cdot 2^n \\ &= 2^n \end{aligned}}$$

$$\downarrow$$

$$\boxed{\begin{aligned} 2^{100} - 2^{99} &= 2^{1+99} - 2^{99} \\ &= 2^1 \cdot 2^{99} - 2^{99} \\ &= 2 \cdot 2^{99} - 1 \cdot 2^{99} \\ &= (2-1) \cdot 2^{99} \\ &= 1 \cdot 2^{99} \\ &= 2^{99} \end{aligned}}$$

$2^{99+1} - 2^{99} = 2^{99}$ holds "for any value of 99"...
We are going to see how to do something like this
for categories (esp. hyperdoctrines, $\approx$ toposes),
using "dictionary tricks".

How generalizations work?
How do we think?
What are the possible/usual roles of diagrams?
Which kinds of reasonings are closer to intuition?
I discovered something quite surprising at UniLog'2010...
(at the talks of Danielle Macbeth and Juan Luis Gastaldi)

it is time to look for further conceptual tools
in Kant and Frege!

Where we are heading
Fact: every "pure" term that deserves the name

$$(A, B) \mapsto (a, b \mapsto b, a)$$

is the (polymorphic) "flip" function.

More precisely:
we have a procedure, $T \mapsto P_T$, that produces
for each "pure" type T (a term is *pure*
when it has no constants besides the sorts)
a property P_T that every pure term $t : T$ obeys.

This is called *parametricity*.
(A good introductory reference:
Phil Wadler's "*Theorems for Free*", 1988).

All the categorical constructions that I will show are "pure",
and it must be possible to use tools from parametricity to
prove meta-theorems about properties of pure constructions.

However, these tools (from parametricity)
are hard to understand, especially if we are
outside the Category Theory/λ-calculus communities,
without much acess to the "oral culture" of these areas.

Why topos theory (is important)
Models:
• Sheaves form (Grothendieck) toposes
• Every $\mathbf{Set}^{\mathcal{U}}$ is a topos (← Non-Standard Analysis)
• There are toposes with nilpotent infinitesimals (← SDG)
• There are toposes with polymorphism and parametricity
• There are toposes in which everything is constructive/computable

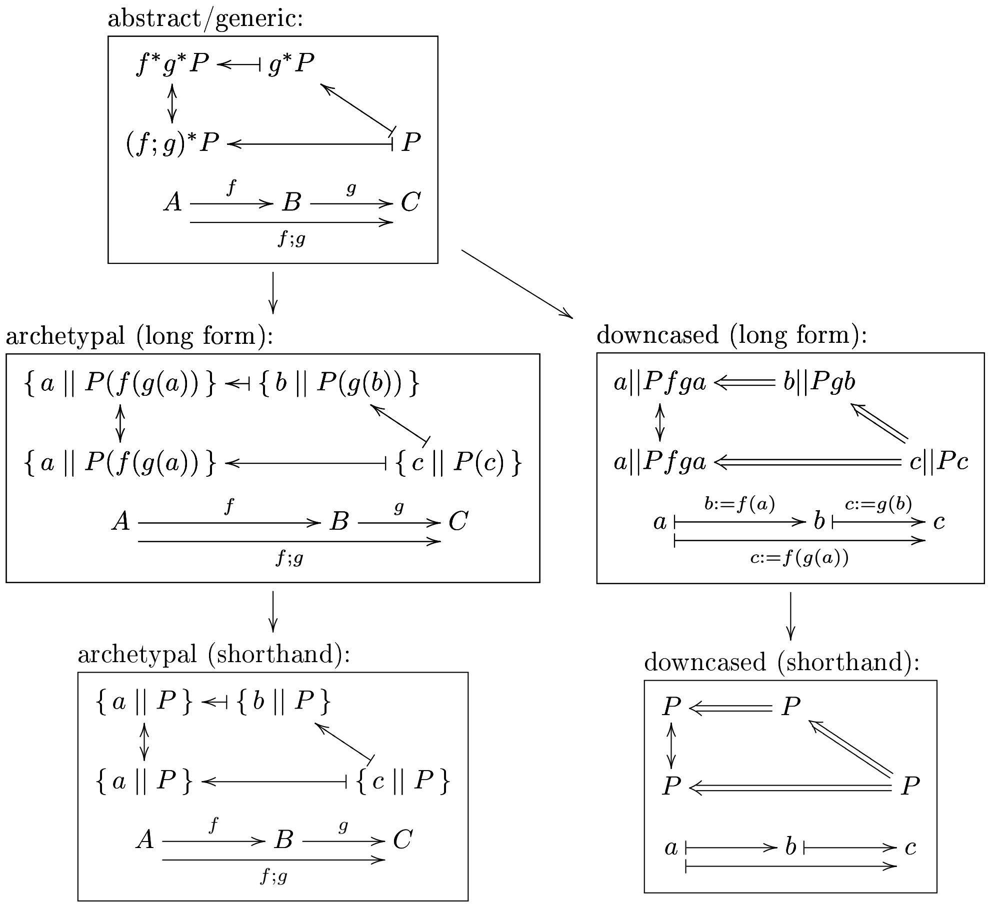
By realizing that a certain category **C** is
(or can be enlarged to) a topos
we immediately *know* a lot about it...
We can carry a lot of our knowledge about **Set**
(constructions, properties) to **C**,
and important constructions in **C** may turn out
to correspond to something simple and well-known in **Set**...

Topos theory (and Categorical Semantics in general)
is about *translating knowledge*
and *recycling theorems*.

And this point is becoming more and more important:
most *proof assistants* are based on
(intuitionistic) type theories whose models
do not fit well in ZFC
(Reynolds 1984: "Polymorphism is not Set-Theoretic")
and learning Topos Theory is (perhaps)
one of the best ways to make all this make sense...

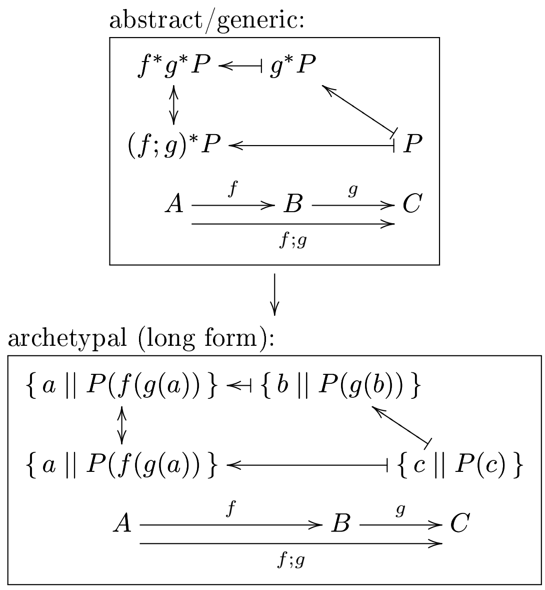
Lifting basic topos theory
Here's a conjecture, whose *technical* details
should not be deep:
it should be possible to develop and prove
almost all of the basic theory of toposes, modalities,
sheaves and geometrical morphisms
in the archetypical cases (i.e., in '$\mathbf{Set}^{\mathbb{D}}$'s),
where everything is very concrete,
and then lift all the constructions and proofs
to the general case.

Downcasing diagrams



The ideas from the method of "downcasing types"
are often used in disguise...
The way of structuring diagrams
(both for the generic case, in abstract notation,
and for the archetypal case — plus its shorthands)
comes from what has to be done to make the corresponding diagrams
make sense in downcased notation.
The dictionary tricks — which include
making '$P \mapsto g^*P$' stand for a functor,
and putting corresponding diagrams side to side —
also come from downcased notation.
(Also the ideas like lifting, etc.)

Downcasing diagrams (2)



Also, the abstract/generic case often makes distinctions
that disappear when we look at the archetypical case...
For example, the archetypal fibration, CanSub(**Set**),
is *split*, and so $f^*g^*\{\, c \mid\mid P \,\} = (f;g)^*\{\, c \mid\mid P \,\}$;
in the generic case there's a canonical iso $f^*g^*P \leftrightarrow (f;g)^*P$,
but it doesn't need to be the identity.

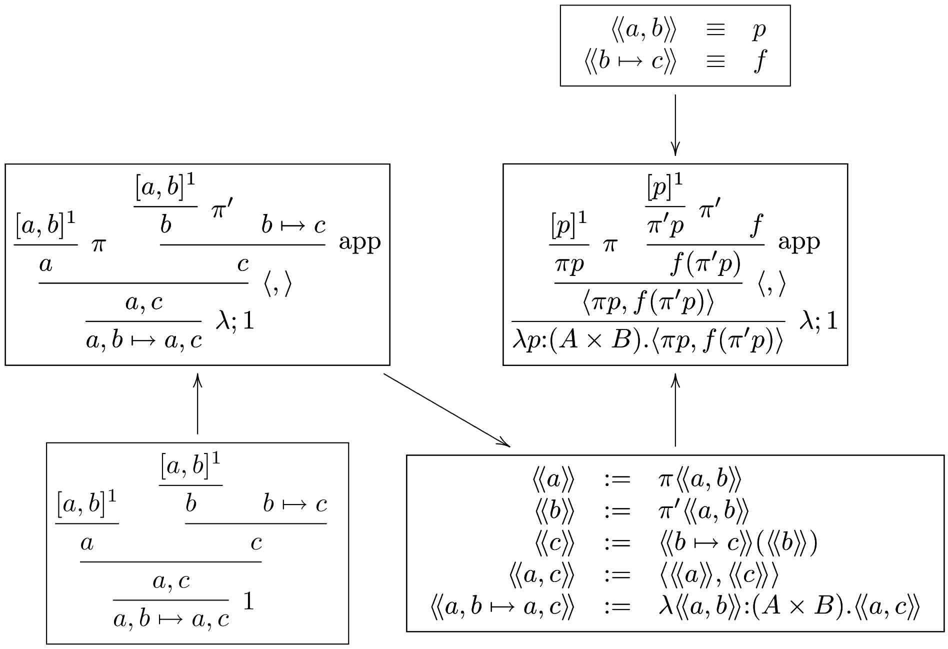
If we are *trying to* model categorically
a certain archetypical case (by generalizing it "in the right way")
then *certain* equalities in the archetypical case
should become at least canonical isos in the generic case...
in CCCs and hyperdoctrines "one half" (i.e., one direction)
of these canonical isos will comes from natural constructions;
what we will do is to *impose* that the these natural constructions
should be invertible.

As we will work with (proto-)*structure* instead of with *properties*
we will require "proto-inverses" instead of requiring *invertibility*.

Can we make Topos Theory more accessible?
Can we make Topos Theory more accessible using these techniques?
At the moment, unfortunately, *not really* —
the rules for the **classifier object** are problematic.
However, one of the most interesting ideas in Topos Theory
is how we can interpret first-order logic *inside* a topos —
and this can also be done in hyperdoctrines.

Hyperdoctrines came a few years before toposes.
Hyperdoctrines are exactly the categories where
we can interpret (intuitionistic, typed) first-order logic.
The definition of a hyperdoctrine seems much more convoluted
that the definition of a topos, but that's because the presence
of a classifier object simplifies everything —
if we add a classifier rule to the hyperdoctrine rules
we see that many of the rules that we had before
can be discarded — they become redundant.

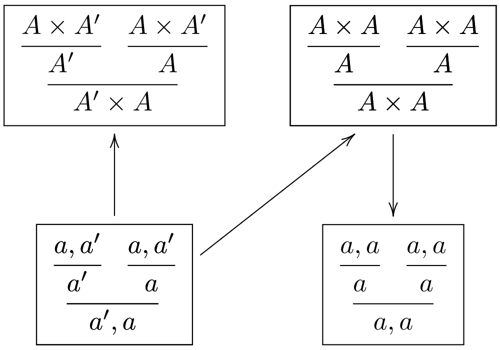
From trees to dictionaries



Each bar of a downcased tree is a *definition*.
If we list all these definitions together we get a "dictionary" that *almost* gives us back the λ-tree...
we need just a few more hints — the standard names of the variables and the constants.

Ambiguities

Liftings are not usually unique...
What should be the type of 'a'? A or A'?
To lift we usually need "hints".
The downcasing that starts with "a, a" below
is (apparently?) valid, but ambiguous.
I am not going to define what a good downcasing is.



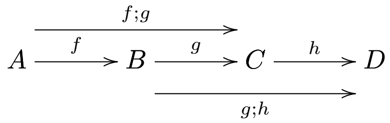
Ambiguities (2)

Composition is not well-behaved syntactically.
One solution (? — or workaround) is to have
a rule 'ren' that renames objects (i.e., create aliases)...

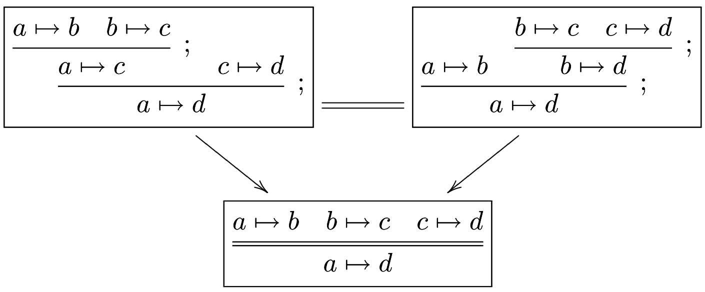
$$\frac{\pi : A \times A \to A \quad f : A \to B}{\pi; f : A \times A \to B} \qquad \frac{a, a' \mapsto a \quad a \mapsto b}{a, a' \mapsto b}$$

$$\frac{\pi' : A \times A \to A \quad f : A \to B}{\pi'; f : A \times A \to B} \qquad \frac{a, a' \mapsto a' \quad a \mapsto b}{a, a' \mapsto b'} \qquad \frac{a, a' \mapsto a' \quad \dfrac{a \mapsto b}{a' \mapsto b'}\,\text{ren}}{a, a' \mapsto b'}$$

Ambiguities (3)
Composition is associative, and that's the reason why we can write $f;g;h$.



Twisting this idea a bit:
we write $f;g;h$ to indicate that $(f;g);h = f;(g;h)$.
We have two different "natural constructions" for things that "deserve the name" $f;g;h$.
Downcasing them, we get:



I used to express that the two "obvious" natural constructions for $(a \mapsto d)$ from $(a \mapsto b)$, $(b \to c)$, $(c \to d)$ give the same result by:

$$\mathrm{wd}\left[\frac{a \mapsto b \quad b \mapsto c \quad c \mapsto d}{a \mapsto d}\right]$$

The entry for that $\langle\!\langle \mathrm{wd}[\ldots] \rangle\!\rangle$ in the dictionary would have to say *which* "obvious constructions" were involved.

More on the 'wd' symbol: the syntactical world
A category $\mathbf{C}$ is a tuple:

$$(\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}}; \mathsf{assoc}_{\mathbf{C}}, \mathsf{idL}_{\mathbf{C}}, \mathsf{idR}_{\mathbf{C}})$$

Where the first four components are "structure"
and the last three are "properties" — equations.
The last three are 'wd' things.

I used to carry all the 'wd's around,
as if they were very very important.
I don't anymore.
It turns out that we can define a projection

$$\frac{(\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}}; \mathsf{assoc}_{\mathbf{C}}, \mathsf{idL}_{\mathbf{C}}, \mathsf{idR}_{\mathbf{C}})}{(\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}})}\ \mathsf{psyn}$$

that drops the equations.
This works for isos, functors, NTs, adjunctions, monads, etc, also.
We can do a "projected version" of Category Theory
and then lift the results back to the "Real World"
(not automatically — many steps have to be done by hand — but whatever).
I call the "projected version" the "Syntactical World".
The projection keeps the "syntactical part" — the "structure" —
and drops the "equational part" — the "properties".

The "projection on the syntactical world", psyn,
is the main operation that we are studying here.
Warning: it will often be used implicitly!
Warning 2: psyn can be defined in many different,
slightly incompatible ways, so before studying its meta-properties
we need to collect enough non-trivial examples of it at work...

Each projection of a non-trivial categorical theorem, T
into the syntactical world, $T^- := \mathsf{psyn}(T)$,
gives a non-trivial example of a lifting:
T^- lifts to T.

The syntactical world: proto-things
I use the prefix "proto" to refer to the projected structures.
A *proto-category* is a 4-uple $(\mathbf{C}_0, \mathrm{Hom}_\mathbf{C}, \mathrm{id}_\mathbf{C}, \circ_\mathbf{C})$.
A *proto-functor* $F : \mathbf{A} \to \mathbf{B}$ is a 2-uple: (F_0, F_1)
(an "action on objects" and an "action on morphisms" —
the equational components, respcomp and respids, have been dropped).

Sometimes there are several possible "design choices"
for a proto-structure...
I will just say which definitions of proto-things work best.
(No time for the full rationale today!)

A *proto-inverse* for a morphism $f : A \to B$
is a just a morphism $f^{-1} : B \to A$.
Note that we don't have the conditions $f; f^{-1} = \mathrm{id}_A$ and $f^{-1}; f = \mathrm{id}_B$,
so the name f^{-1} may be a bit misleading.
A *proto-iso* is a pair (f, f^{-1}), where f^{-1}
is a proto-inverse for f.

A *proto-natural transformation* ("proto-NT") $T : F \to G$
is just an operation $A \mapsto (FA \overset{T_A}{\to} GA)$
(we drop the "square condition" that says that for each
$f : A \to B$ the "obvious square" must commute).

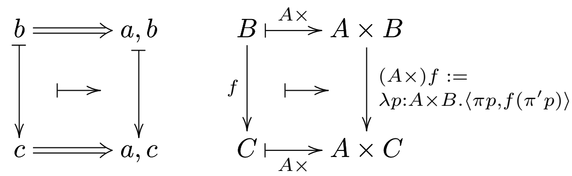
A *proto-inverse* for a (proto-)natural transformation $T : F \to G$
is just a (proto-)natural transformation $T^{-1} : G \to F$ —
for each object A the morphisms TA and $T^{-1}A$ are the two
directions of a proto-iso.
A *proto-natural isomorphism* is a proto-NT plus a
proto-inverse for it.

What are proto-products, proto-exponentials, proto-fibrations...?

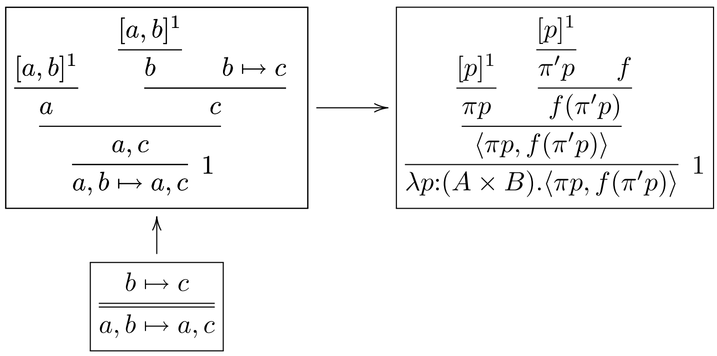
Downcasing functors

Fix a set A.
Let $(A\times) : \mathbf{Set} \to \mathbf{Set}$ denote the functor
that takes each set B to $A \times B$.

"*The*" functor?
In some contexts the action on morphisms, $(A\times)_1$,
is "obvious" once the action on objects, $(A\times)_0$, is given.
How?



By lifting:



(a double bar is like a folded accordion)

Adjunctions

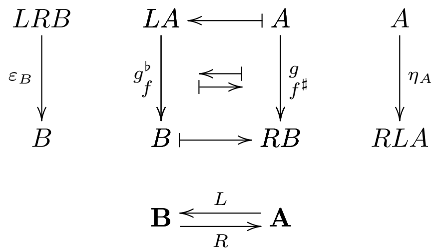
If L and R are functors going in opposite directions between two categories, say,

$$\mathbf{B} \underset{R}{\overset{L}{\leftrightarrows}} \mathbf{A}$$

then a *proto-adjunction*, $L \dashv R$, is an 8-uple,

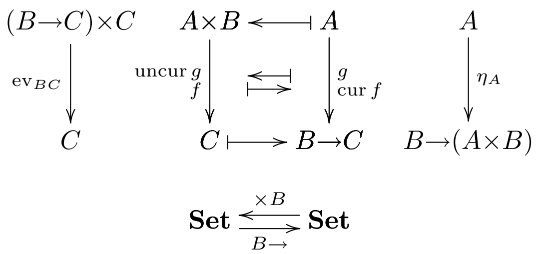
$$(\mathbf{A}, \mathbf{B}, L, R, \flat, \sharp, \eta, \epsilon)$$

that we draw as:



There is a lot of redundancy in this definition...
Most operations can be reconstructed from the others.

Our archetypal adjunction will be this one:
(not the usual $F \dashv U$!)

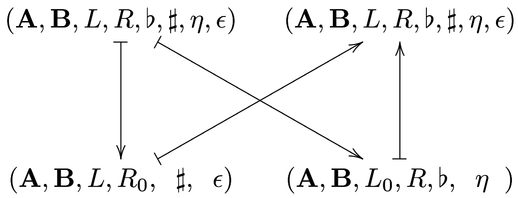


Cheap and expensive adjunctions

In the supermarket you can buy DeLuxe adjunctions,
that are expensive but come with all the bells and whistles,
and you can buy the cheap, minimal, economy models,
that come in kit form, and (apparently) do much less...

There are theorems that take cheap adjunctions and
produce expensive adjunctions from them — for free.

In the Real World cheap and expensive adjunctions
are equivalent: if you do



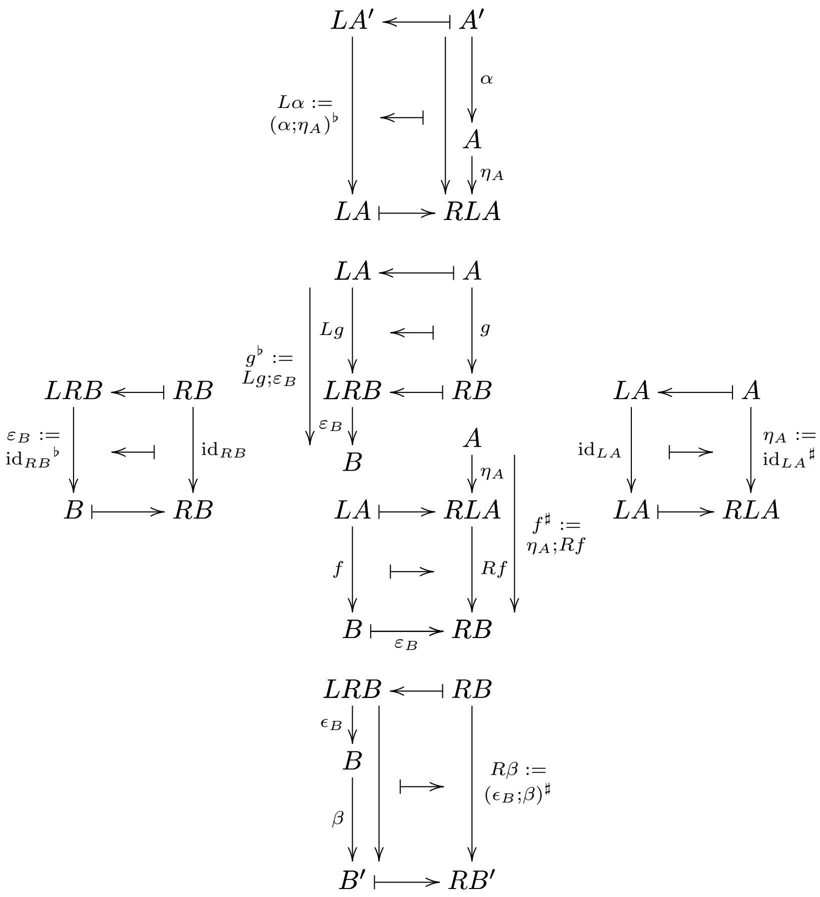
you get the original expensive adjunction back.
In the Syntactical World you may get something different,
but that doesn't matter.

Cheap and expensive adjunctions (2)

An "expensive" proto-adjunction $L \dashv R$ is an 8-uple:

$$(\mathbf{A}, \mathbf{B}, L, R, \flat, \sharp, \eta, \epsilon).$$

Here is how to reconstruct some of its components from the other ones.



Programming with long names

```
⟪a,b|->a,c⟫ = function (⟪a,b⟫)
    local a = pi1(⟪a,b⟫)
    local b = pi2(⟪a,b⟫)
    local c = ⟪b|->c⟫(b)
    local ⟪a,c⟫ = {a, c}
    return ⟪a,c⟫
  end
```

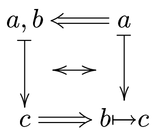
→

```
dnc00 = function (dnc01)
    local a = pi1(dnc01)
    local b = pi2(dnc01)
    local c = dnc02(b)
    local dnc03 = {a, c}
    return dnc03
  end
```

↓

```
⟪a,b ↦ a,c⟫ = function (⟪a,b⟫)
    local a = pi1(⟪a,b⟫)
    local b = pi2(⟪a,b⟫)
    local c = ⟪b ↦ c⟫(b)
    local ⟪a,c⟫ = {a, c}
    return ⟪a,c⟫
  end
```

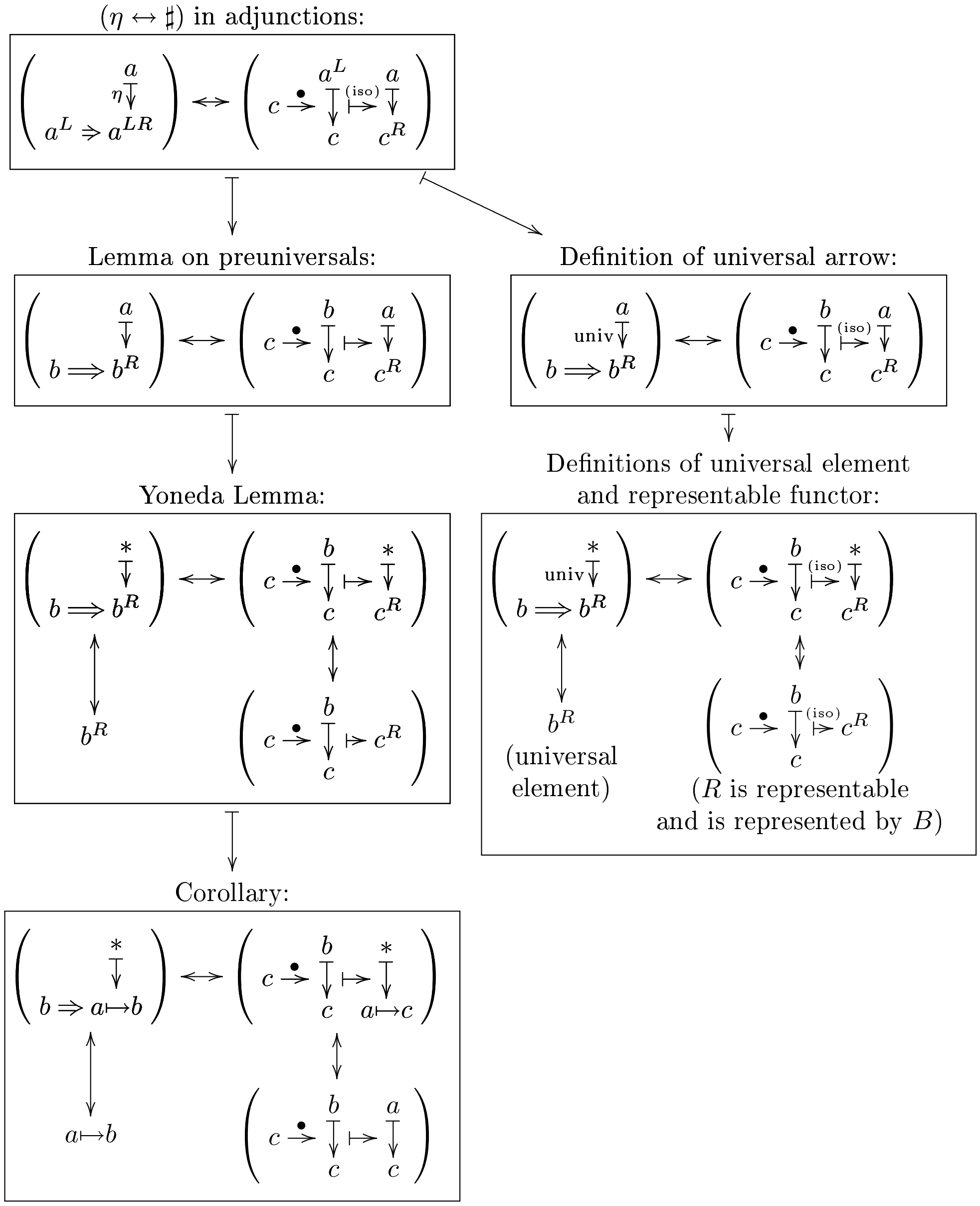
Note that diagrams are perfectly good as names.
We can have:

⟪  ⟫ `:= {...}`

In the (ascii) source code this would be:

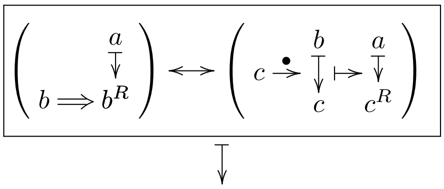
⟪`\diag{adjunction1}`⟫ `:= {...}`

The Yoneda Lemma

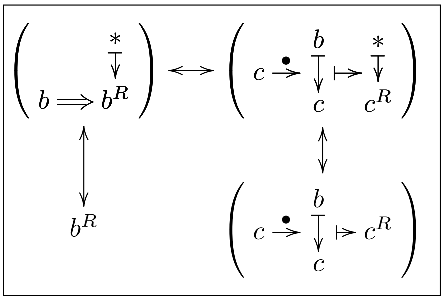


The Yoneda Lemma (2)
We can formalize the previous diagram in type system (or in a programming language — say, ML or Coq). Let's look at a miniature.

Lemma on preuniversals:



Yoneda Lemma:



In the top box:
$B : \mathbf{B}_0$
$A : \mathbf{A}_0$
$R : \mathbf{B} \to \mathbf{Set}$
$a \mapsto b^R \equiv g : \mathrm{Hom}_{\mathbf{A}}(A, RB)$

$$\left(\begin{array}{ccc} & & a \\ & & \downarrow \\ b & \Longrightarrow & b^R \end{array}\right) \equiv (A, B, g)$$

$c \Rightarrow (b \mapsto c) \equiv \mathrm{Hom}_{\mathbf{B}}(B, -) : \mathbf{B} \to \mathbf{Set}$
$c \Rightarrow (a \mapsto c^R) \equiv C \mapsto \mathrm{Hom}_{\mathbf{A}}(A, RC) \equiv \mathrm{Hom}_{\mathbf{A}}(A, R-) : \mathbf{B} \to \mathbf{Set}$

$$\left(\begin{array}{ccccc} & & b & & a \\ c & \overset{\bullet}{\to} & \downarrow & \mapsto & \downarrow \\ & & c & & c^R \end{array}\right) \equiv T : \mathrm{Nat}(\mathrm{Hom}_{\mathbf{B}}(B, -), \mathrm{Hom}_{\mathbf{A}}(A, R-))$$

$g \leftarrow\!\shortmid\, T := \lambda T.(TB\mathrm{id}_B)$
$g \mapsto T := \lambda f.(\lambda C.\lambda g.(f; Rg))$

When we move to the bottom box we specialize:
$\mathbb{A} := \mathbf{Set}$
$A := 1 \equiv \{*\}$
$* : 1$

Monads

A *protomonad* for a proto-endofunctor $T : \mathbf{A} \to \mathbf{A}$ is a 4-uple:

$$(\mathbf{A}, T, \eta, \mu)$$

that we draw as:

$$A \xrightarrow{\eta_A} TA \xleftarrow{\mu_A} TTA$$

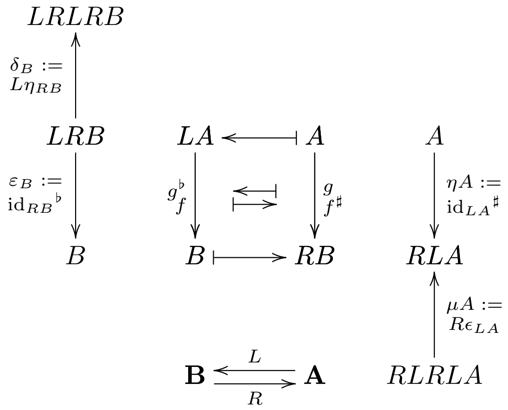
A *proto-comonad* for a proto-endofunctor $S : \mathbf{B} \to \mathbf{B}$ is a 4-uple:

$$(\mathbf{B}, S, \varepsilon, \delta)$$

that we draw as:

$$B \xleftarrow{\varepsilon_B} SB \xrightarrow{\delta_B} SSB$$

Each proto-adjunction induces both a proto-monad and a proto-comonad. We draw all these together as:



$$\cfrac{\cfrac{LA}{\epsilon_{LA} : LRLA \to LA}\ \epsilon}{\mu_A := R\epsilon_{LA} : RLRLA \to RLA}\ R \qquad \cfrac{\cfrac{RB}{\eta_{RB} : RB \to RLRB}\ \sharp}{\delta_B := L\eta_{RB} : LRB \to LRLRB}\ L$$

From monads to adjunctions
An expensive adjunction comes with a monad.
If we erase the '♭', the '♯', etc of this adjunction
and leave only the monad, can we reconstruct
the original adjunction from that?
The answer is **no**.
But we can construct two adjunctions from the monad,

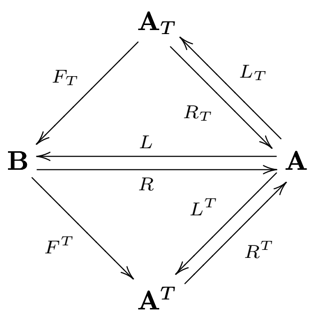
$\mathbf{A}_T \underset{R_T}{\overset{L_T}{\rightleftarrows}} \mathbf{A}$ (Kleisli)

$\mathbf{A}^T \underset{R^T}{\overset{L^T}{\rightleftarrows}} \mathbf{A}$ (Eilenberg-Moore)

that are related to the original adjunction

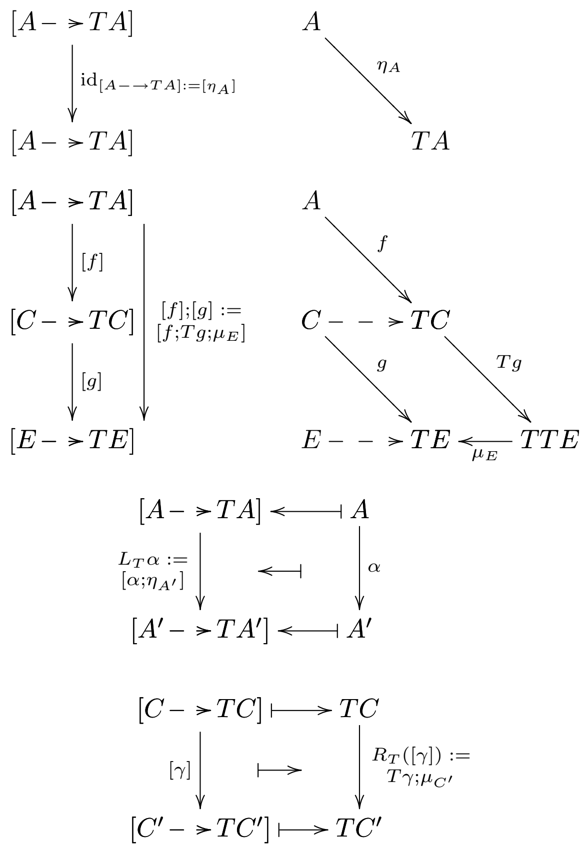
$\mathbf{B} \underset{R}{\overset{L}{\rightleftarrows}} \mathbf{A}$

in interesting ways —
by comparison functors F_T and F^T.



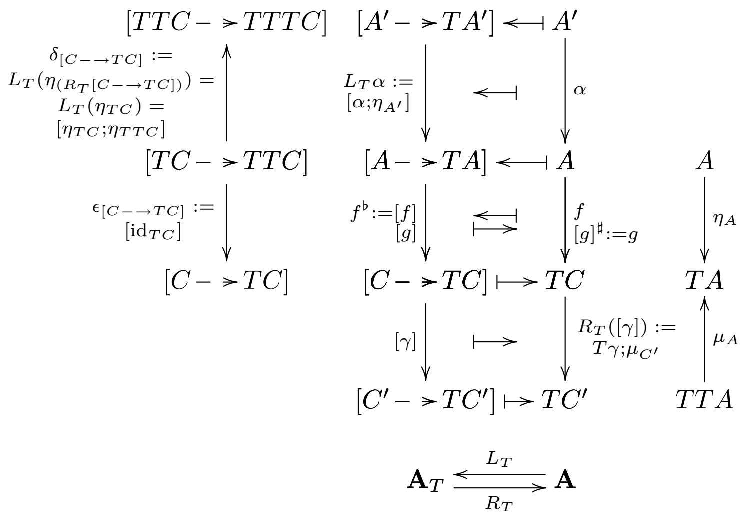
The Kleisli category of a monad

$\mathbf{A}_T$ has the same objects as $\mathbf{A}$,
but we write them in a funny way —
$[A \dashrightarrow TA]$ instead of A —
because $\mathrm{Hom}_{\mathbf{A}_T} \neq \mathrm{Hom}_{\mathbf{A}}$ and $\circ_{\mathbf{A}_T} \neq \circ_{\mathbf{A}}$.
$[f] : [A \dashrightarrow TA] \to [C \dashrightarrow TC] \quad$ (in $\mathbf{A}_T$)
is $f : A \to TC$ in $\mathbf{A}$.
$[f]; [g] := [f; Tg; \mu_E]$.



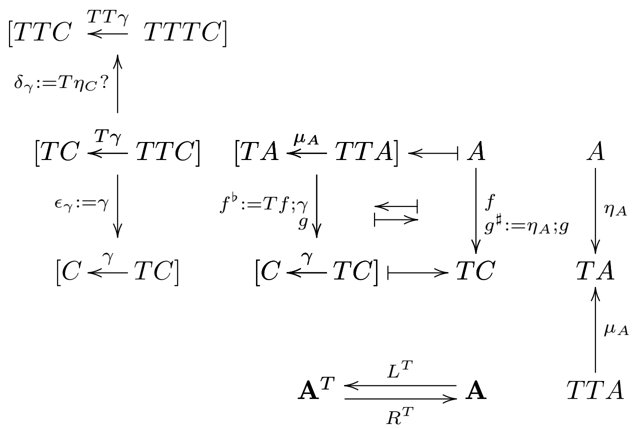
Proving that $\circ_{\mathbf{A}_T}$ is non-trivial, by the way —
but in the syntactical world that doesn't matter.

The Kleisli adjunction of a monad



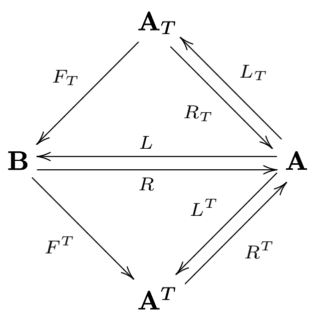
The Eilenberg-Moore category of a monad
A *proto-algebra* for a monad $(\mathbf{A}, T, \eta, \mu)$ is a pair $(A,\ \alpha : TA \to T)$.
An *algebra* for a monad $(\mathbf{A}, T, \eta, \mu)$ is a proto-algebra (A, α) that obeys $T\alpha; \alpha = \mu_A; \alpha$.
A *proto-morphism* $f : (A, \alpha) \to (C, \gamma)$ of (proto-)algebras is just a morphism $f : A \to C$.
A *morphism* of algebras is a proto-morphism f that obeys $\alpha; f = Tf; \gamma$.
The *proto-Eilenberg-Moore category* of a monad $(\mathbf{A}, T, \eta, \mu)$ has the proto-algebras as objects and the proto-morphisms of (proto-)algebras as morphisms. We write it as $\mathbf{A}^T$.
The *Eilenberg-Moore category* of a monad $(\mathbf{A}, T, \eta, \mu)$, $\mathbf{A}^T$, has the algebras as objects and the morphisms of algebras as morphisms. We also write it as $\mathbf{A}^T$,

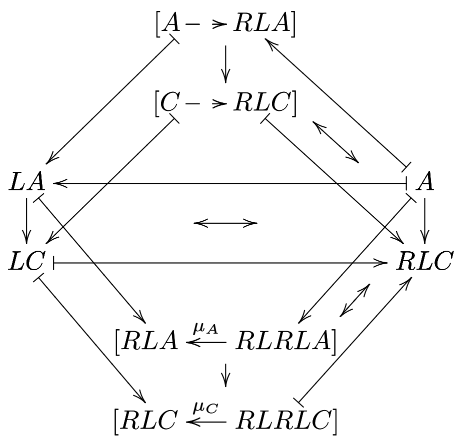
Here is the adjunction $\mathbf{A}^T \underset{R^T}{\overset{L^T}{\leftrightarrows}} \mathbf{A}$:



The comparison theorem

Everything fits in this diagram:





$$\cfrac{\cfrac{[g] : [A - \twoheadrightarrow RLA] \to [C - \twoheadrightarrow RLC]}{g : A \to RLC}\ \text{ren}}{F_T([g]) := g^\flat : LA \to LC}\ \flat$$

$$\cfrac{\cfrac{\cfrac{A}{LA}\ L}{\epsilon_{LA} : LRLA \to LA}\ \epsilon}{\mu_A = R\epsilon_{LA} : RLRLA \to RLA}\ R \qquad \cfrac{\cfrac{B}{\epsilon_B : LRB \to B}\ \epsilon}{R\epsilon_B : RLRB \to RB}\ R$$

Beck's lemma
From Beck's thesis (1967), reprinted at:
`http://www.tac.mta.ca/tac/reprints/articles/2/tr2abs.html`

> DEFINITION 3. The adjoint pair $\alpha : F \dashv U$ is tripleable if $\Phi : B \to A^T$ is an equivalence of categories.
>
> DEFINITION 3'. A functor $U : B \to A$ is tripleable if U has a left adjoint F and the adjoint pair $F \dashv U$ is tripleable.
>
> (...)
>
> THEOREM 1. Let $\alpha : F \dashv U$ be an adjoint pair.
>
> (1) If B has coequalizers, then there exists a left adjoint $\hat{\Phi} \dashv \Phi$.
>
> Assuming the existence of $\hat{\Phi}$:
>
> (2) If U preserves coequalizers, then the unit of $\hat{\Phi} \dashv \Phi$ is an isomorphism $AT \overset{\equiv}{\to} \hat{\Phi}\Phi$.
>
> (3) If U reflects coequalizers, then the counit is an isomorphism $\Phi\hat{\Phi} \overset{\equiv}{\to} B$.
>
> Finally, in the presence of (2), (3) can be replaced by:
>
> (3') If U reflects isomorphisms, then the counit is an isomorphism $\Phi\hat{\Phi} \overset{\equiv}{\to} B$.

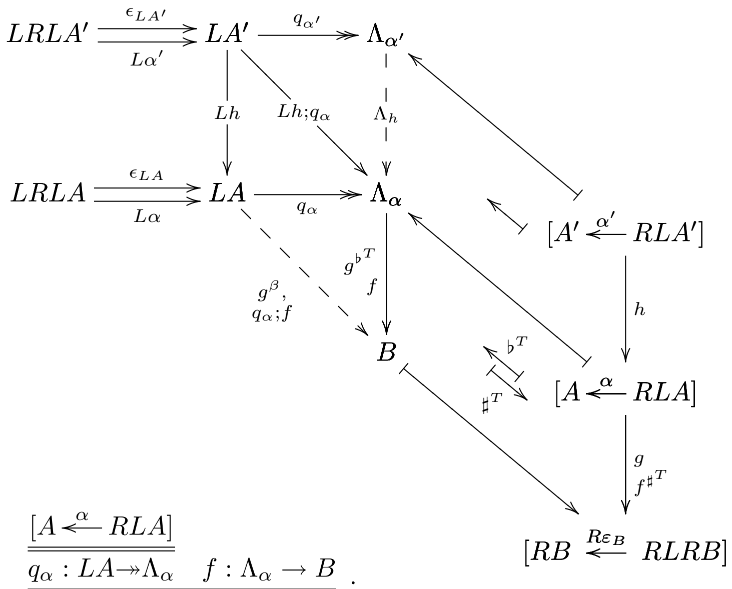
I will call item (1) of Theorem 1 "Beck's Lemma".
In our notation this will be:
if in an adjunction $\mathbf{B} \underset{R}{\overset{L}{\leftrightarrows}} \mathbf{A}$
the category $\mathbf{B}$ has coequalizers then
the comparison functor $F^T : \mathbf{B} \to \mathbf{A}^T$
has a left adjoint, Λ (i.e. we have an adjunction $\Lambda \dashv F^T$).
We need to construct Λ_0, Λ_1, $\flat^T$, $\sharp^T$.

Beck's lemma (2)
Sketch of the proof (a construction, actually):
Let's will write Λ as $[A \xleftarrow{\alpha} TA] \mapsto \Lambda_\alpha$.

The action of Λ on objects:
for each object $[A \xleftarrow{\alpha} TA]$ in $\mathbf{A}^T$ we need to produce an object Λ_α in $\mathbf{B}$.
We define it as the coequalizer of ϵ_{LA} and $L\alpha$.

$$\dfrac{\dfrac{\dfrac{A}{LA}\;L}{\epsilon_{LA} : LRLA \to LA}\;\epsilon \qquad \dfrac{\dfrac{[A \xleftarrow{\alpha} TA]}{\alpha : RLA \to A}\;\text{ren}}{L\alpha : LRLA \to LA}\;L}{\dfrac{q_\alpha : LA \twoheadrightarrow \Lambda_\alpha}{\Lambda_\alpha}\;\text{tgt}}\;\text{coeq}$$

The action of Λ on morphisms ($\Lambda_h := (Lh; q_\alpha)/q_{\alpha'}$), and the transpositions:



$$\dfrac{\dfrac{\dfrac{[A \xleftarrow{\alpha} RLA]}{q_\alpha : LA \twoheadrightarrow \Lambda_\alpha} \quad f : \Lambda_\alpha \to B}{q_\alpha; f : LA \to B}\;;}{f^{\sharp^T} \;:=\; (q_\alpha; f)^\sharp : A \to RB}\;\sharp$$

$$\dfrac{\dfrac{[A \xleftarrow{\alpha} RLA]}{q_\alpha : LA \twoheadrightarrow \Lambda_\alpha} \qquad \dfrac{\dfrac{g : [A \xleftarrow{\alpha} RLA] \to [RB \xleftarrow{R\varepsilon_B} RLRB]}{g : A \to RB}}{g^\flat : LA \to B}\;\flat}{g^{\flat^T} \;:=\; g^\flat / q_\alpha : \Lambda_\alpha \to B}\;\text{fact.through.coeq}$$

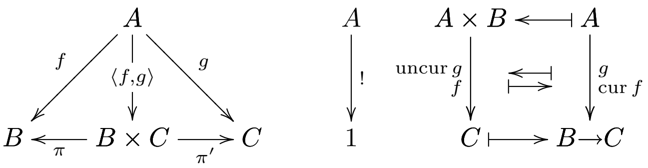
Monads and cohomology
Quoting again from Beck's thesis (p.12)...

> **2. Cohomology**
>
> Adjoint functors, it is now well known, lead to cohomology ([Eilenberg & Moore (1965a), Godement (1958), Mac Lane (1963)] — or to homotopy [Huber (1961)]). If
>
> $$\underline{A} \xrightarrow{F} \underline{B} \xrightarrow{A} \underline{A} \qquad (F \dashv U)$$
>
> is an adjoint pair, objects of the form $AF \in |B|$ are regarded as *"free" relative to the underlying object functor* U. The counit
>
> $$XUF \xrightarrow{X\epsilon} X$$
>
> is intuitively the first step of a functorial free resolution of any object $X \in |\underline{B}|$. By iterating UF one extends X to a *free simplicial resolution* of X, and defines derived functors as usual in homological algebra. Here we only consider the simplest case, that of defining *cohomology groups*
>
> $$H^n(X, Y), \qquad n \geq 0,$$
>
> *of an object* $X \in |\underline{B}|$ *with coefficients in an abelian group object* $Y \in |\underline{B}|$, *relative to the given underlying object functor* $U : \underline{B} \to A$ (having a left adjoint). Tripleableness of $F \dashv U$ will not play any appreciable role until we discuss special properties of the cohomology in §3. We now recall the details of the construction of the cohomology groups. Some of the terms used are clarified in the proof of Theorem 2, which summarizes the main properties the cohomology possesses.

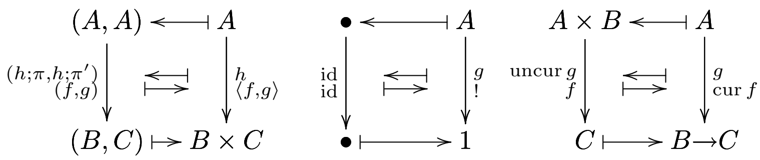
How much of that has nice syntactical proofs?
(I don't know yet!)

Cartesian Closed Categories

a *Cartesian Closed Category* ("CCC") $(\mathbf{C}, \times, 1, \to)$
is a category $\mathbf{C}$ plus a "cartesian closed structure" $(\times, 1, \to)$ on it.
Again, we will have cheap ways and expensive ways to specify $(\times, 1, \to)$.
The cheap ways (there will be several of them)
will appear from formalizing this diagram:

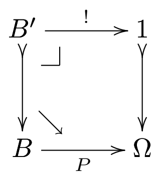


Another way — the "adjoint presentation" of a CCC —
will be by requiring that the functors $A \mapsto (A \times A)$,
$A \mapsto \bullet$, and $A \mapsto A \times B$ (note that for each object B of $\mathbf{C}$
we have a different $A \mapsto A \times B$) all have right adjoints:



The adjoint presentation is quite elegant, but if we
expand all the details we see that it is much more expensive
than the other ones.

By the way: a (cheap) topos, $(\mathbf{C}, \times, 1, \to, \Omega)$,
is a CCC plus a "classifier object", Ω,
where the Ω obeys this magic axiom,
that has *lots* of consequences:



Cheap and expensive toposes
What matters to us is:
a CCC is a category in which we can interpret λ-calculus;
a topos is a category in which we can interpret (a kind of) set theory;
a hyperdoctrine is a category in which we can interpret first-order logic.
(I'm simplifying things a little bit, but anyway).

When we buy a cheap topos it takes a lot of work
to build the constructions that interpret first-order logic in it...
in a hyperdoctrine they sort of come out-of-the-box.

An expensive topos has all the structure and
properties of an expensive hyperdoctrine
plus a few more.

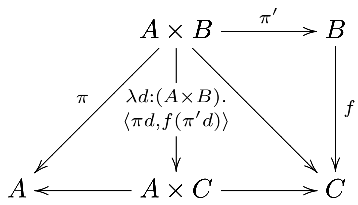
The category of sets, **Set**, is a topos, and therefore
if you go to the supermarket with the intent of buying
the category of sets you will find it there in several
different presentations... the more expensive ones
have all the hyperdoctrine rules, all the topos rules,
plus some.

λ-calculus in a CCC: an example

We can interpret the λ-construction

$$\frac{A \quad f : B \to C}{\lambda d{:}(A \times B).\langle \pi d, f(\pi' d)\rangle : A \times B \to A \times C}$$

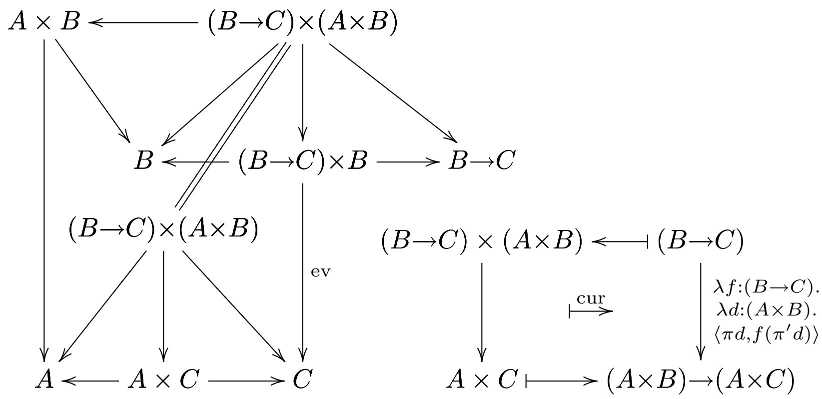
in a Cartesian Closed Category as this diagram:



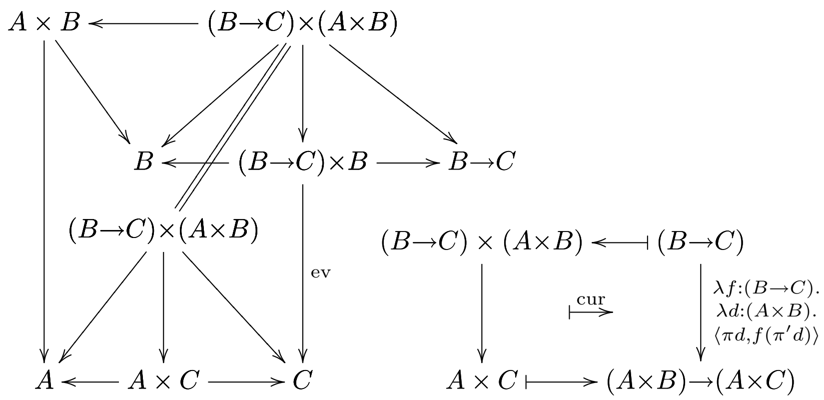
And we can interpret its "internalization",

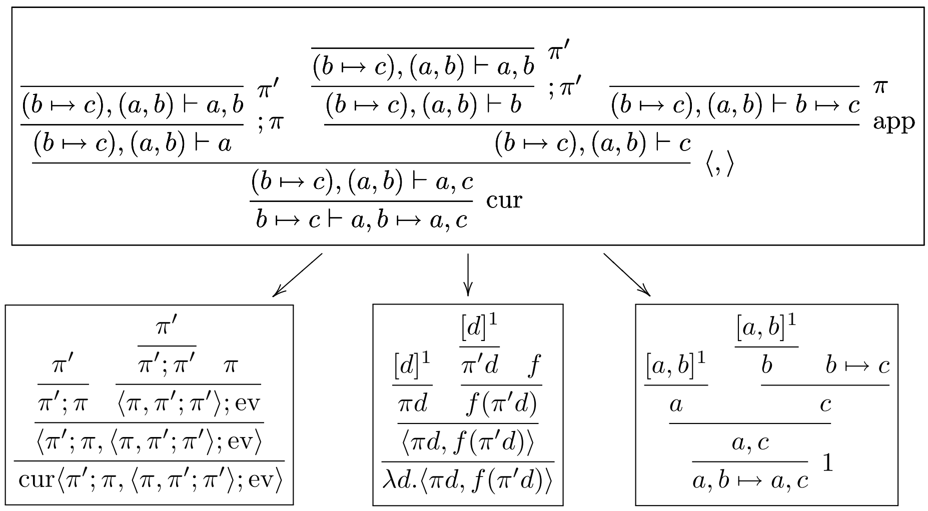
$$\frac{A \quad B \quad C}{\lambda f{:}(B{\to}C).\lambda d{:}(A \times B).\langle \pi d, f(\pi' d)\rangle : (B \to C) \to (A \times B \to A \times C)}$$

using this diagram:



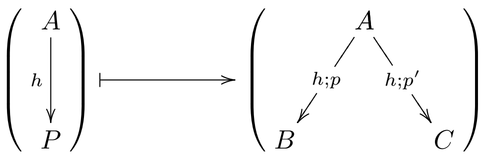
λ-calculus in a CCC: an example (2)
...but in a case like this the diagrams
are secondary, and for most λ-terms they are
unbearably big... we should focus on the trees.





The product property

For any diagram of the form $B \xleftarrow{p} P \xrightarrow{p'} C$ in a category $\mathbf{C}$
we have a natural operation,



$$\begin{array}{rcl} \mathrm{Hom}_{\mathbf{C}}(A,P) & \longrightarrow & \mathrm{Hom}_{\mathbf{C}}(A,B) \times \mathrm{Hom}_{\mathbf{C}}(A,C) \\ h & \longmapsto & (h;p,\ h;p') \end{array}$$

whose action is to compose any given $h : A \to P$ with p and p'.
Both $\mathrm{Hom}_{\mathbf{C}}(-,P)$ and $\mathrm{Hom}_{\mathbf{C}}(-,B) \times \mathrm{Hom}_{\mathbf{C}}(-,C)$ are
(contravariant) functors from $\mathbf{C}^{\mathrm{op}}$ to **Set**,
and the operation above is a natural transformation

$$\mathsf{prod}^{\natural} : \mathrm{Hom}_{\mathbf{C}}(-,P) \to \mathrm{Hom}_{\mathbf{C}}(-,B) \times \mathrm{Hom}_{\mathbf{C}}(-,C)$$

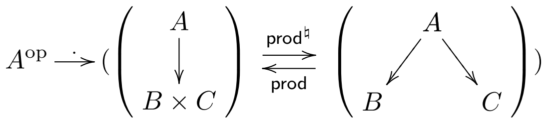
The *product property* for a diagram $B \xleftarrow{p} P \xrightarrow{p'} C$.
is an inverse for the "natural" natural transformation above.

Product diagrams

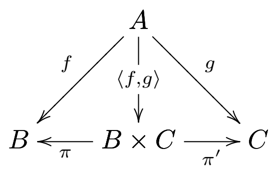
Whenever we write $B \xleftarrow{\pi} B \times C \xrightarrow{\pi'} C$,
using the product symbol in the middle object, it will be implicit that the $B \times C$ must "deserve its name": that is, we must have "projection maps" π and π' (here they were shown explicitly),

and the diagram $B \xleftarrow{\pi} B \times C \xrightarrow{\pi'} C$
must have the product property.

We can draw the two natural transformations together as:



but note that the projection maps did not appear in the picture...
As we can make diagram stand for whatever we want
(because diagrams are valid as long names)
we will take diagrams of this form as meaning:



a diagram $B \xleftarrow{\pi} B \times C \xrightarrow{\pi'} C$,
plus the product property for it,
plus a syntactical cue for how to name the results of prod;
in this case, this is: :

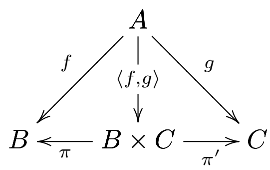
$$\cfrac{\cfrac{f : A \to B \quad g : A \to C}{(f, g) : \mathrm{Hom}_{\mathbf{C}}(A, B) \times \mathrm{Hom}_{\mathbf{C}}(A, C)}}{\langle f, g \rangle : A \to B \times C} \; \mathsf{prod}_A$$

or, more briefly:

$$\frac{f : A \to B \quad g : A \times C}{\langle f, g \rangle : A \to B \times C} \; \mathsf{prod}$$

although this will be an abuse of language, in a sense —

Product diagrams (2)



Note that if we write the rule prod as:

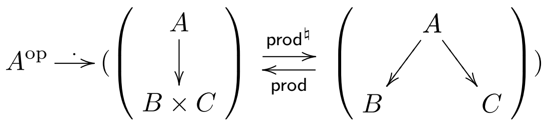
$$\frac{(f : A \to B,\ g : A \to C)}{\langle f, g\rangle : A \to B \times C}\ \textsf{prod}$$

then it is exactly the inverse of:

$$\frac{h : A \to B \times C}{(h;\pi : A \to B,\ h;\pi : A \to C)}\ \textsf{prod}^\natural \quad := \quad \frac{\dfrac{h : A \to B \times C}{h;\pi : A \to B} \quad \dfrac{h : A \to B \times C}{h;\pi : A \to C}}{(h;\pi : A \to B,\ h;\pi : A \to C)}\ \langle,\rangle$$

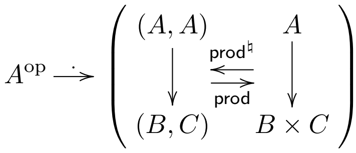
We will repeat this pattern all over the place:
for some derived rule, $\textsf{blah}^\natural$, an inverse blah.

Note that if you were given just $B \xleftarrow{\pi} B \times C \xrightarrow{\pi'} C$ and

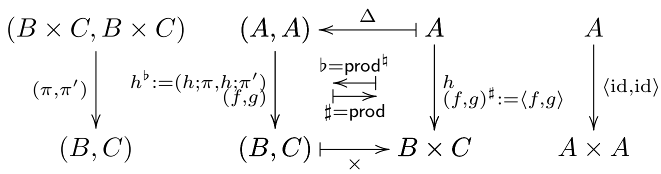


then you would have to figure out the "natural construction"
for $\textsf{prod}^\natural$ yourself; and then prod would be its inverse.

Products as adjunctions
If we rewrite the previous NT using the category $\mathbf{C} \times \mathbf{C}$
and we flip the positions of the two vertical arrows
it becomes:

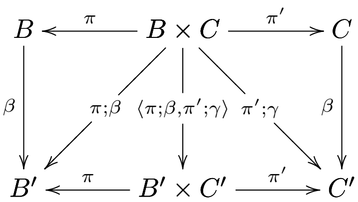


which looks almost like an adjunction...
In fact, if for any two objects B and C of $\mathbf{C}$
we have a product object $B \times C$ (that "deserves its name",
i.e., comes with π, π', and the product property)
then we have this adjunction,

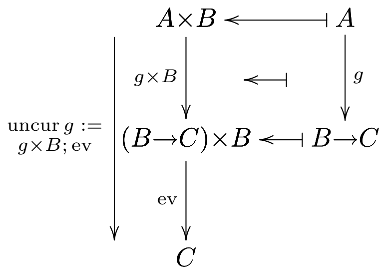


$$\mathbf{C} \times \mathbf{C} \underset{\times}{\overset{\Delta}{\rightleftarrows}} \mathbf{C}$$

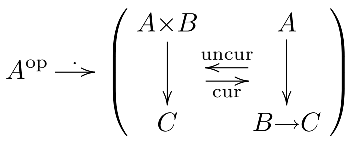
where the action of $\times$ on a morphism $(\beta, \gamma) : (B, C) \to (B', C')$
of $\mathbf{C} \times \mathbf{C}$ yields $\beta \times \gamma := \langle \pi; \beta, \pi'; \gamma \rangle$:



Exponentials
In a category with binary products, $(\mathbf{C}, \times)$,
an object deserves the name $B{\to}C$
if it comes with an evaluation map,
$\mathrm{ev} : (B{\to}C){\times}B \to C$, and an inverse, cur,
for the "uncurrying" operation:

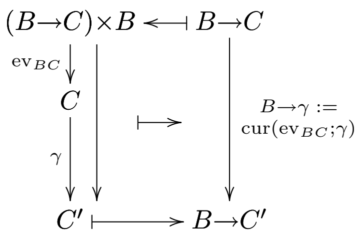


This, again, looks like an adjunction:



where the top functor, $A \mapsto A \times B$, is known.

Let's fix the object B.
If we have the action of $C \mapsto B{\to}C$
on objects — i.e., for each C
we have $B{\to}C$, ev_{BC} and cur_{BC} —
then we can build $(B{\to})_1$.
The trick comes from slide 21 (its bottom rectangle):



We have just built an andjunction $(B\times) \dashv (B{\to})$
from an operation $C \mapsto B{\to}C$.

Exponentials (2)
It turns out that we can also construct
a functor $(B^{\mathrm{op}}, C) \mapsto B{\to}C$.
Its action on objects is obvious;
its action on morphisms takes
each $(\beta^{\mathrm{op}}, \gamma) : (B^{\mathrm{op}}, C) \to (B'^{\mathrm{op}}, C')$
to $(\beta{\to}\gamma) := ({\to}_1)(\beta^{\mathrm{op}}, \gamma) := \mathrm{cur}(\langle \pi, \pi'; \beta\rangle; \mathrm{ev}; \gamma)$,

$$\begin{array}{ccc}
(B^{\mathrm{op}}, C) & \longmapsto & B{\to}C \\
{\scriptstyle (\beta^{\mathrm{op}},\gamma)}\downarrow & & \downarrow{\scriptstyle (\beta\to\gamma) := \; ({\to}_1)(\beta^{\mathrm{op}},\gamma) := \; \mathrm{cur}(\langle\pi,\pi';\beta\rangle;\mathrm{ev};\gamma)} \\
(B'^{\mathrm{op}}, C') & \longmapsto & B'{\to}C'
\end{array}$$

where $\mathrm{cur}(\langle \pi, \pi'; \beta\rangle; \mathrm{ev}; \gamma)$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{B \quad C}{B{\to}C} \quad b' \vdash b}{(b \mapsto c), b' \vdash (b \mapsto c), b} \quad \dfrac{B \quad C}{(b \mapsto c), b \vdash c} \quad c \vdash c'}{(b \mapsto c), b' \vdash c'}}{(b \mapsto c) \vdash (b' \mapsto c')}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{B \quad C}{B{\to}C} \quad \beta : B' \to B}{\langle \pi_{(B\to C)B'}, \pi'_{(B\to C)B'}; \beta\rangle} \quad \dfrac{B \quad C}{\mathrm{ev}_{BC}}\,\mathrm{ev} \quad \gamma : C \to C'}{\langle \pi, \pi'; \beta\rangle; \mathrm{ev}; \gamma}\;;;}{\mathrm{cur}(\langle \pi, \pi'; \beta\rangle; \mathrm{ev}; \gamma)}\,\mathrm{cur}$$

But how can we find a definition
like the one above for $(\beta{\to}\gamma)$ —
without using brute force?

Morphisms as sequents

We can lift the natural deduction tree

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{[b']^1 \quad b' \mapsto b}{b} \quad [b \mapsto c]^2}{c} \quad c \mapsto c'}{c'}}{b' \mapsto c'}\;1}{(b \mapsto c) \mapsto (b' \mapsto c')}\;2$$

to something that looks more like
sequent calculus, in the sense that the hypotheses
are always listed explicitly before the '$\vdash$',
and names with '$\vdash$' stand for morphisms:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{(b \mapsto c), b' \vdash b'}\;\pi' \quad b' \vdash b}{(b \mapsto c), b' \vdash b}\;; \quad \cfrac{}{(b \mapsto c), b' \vdash (b \mapsto c)}\;\pi}{(b \mapsto c), b' \vdash c}\;\text{app} \quad c \vdash c'}{(b \mapsto c), b' \vdash c'}\;;}{(b \mapsto c) \vdash (b' \mapsto c')}\;\text{cur}$$

I don't want to get into the details of this now...

Hyperdoctrines
The following is the precise definition of a hyperdoctrine.
It is too technical, so we will spend the next slides dissecting it.

A hyperdoctrine is a (cloven) fibration, $p : \mathbb{E} \to \mathbb{B}$,
plus some extra structure:

- the base category $\mathbb{B}$ is a CCC
  (i.e., $(\mathbb{B}, \times, 1, \to)$ is a CCC))
- each fiber $\mathbb{E}_B$ is a CCC
  (i.e., each $(\mathbb{E}_B, \land, \top, \supset)$ is a CCC)
- for each map $g : B \to C$ in $\mathbb{B}$
  the change-of-base functor f^* has adjoints $\exists_f \dashv f^* \dashv \forall_f$
- change-of-base preserves $\land$, $\top$, $\supset$ modulo iso
  (P$\land$, P$\top$, P$\supset$)
- the left and right Beck-Chevalley conditions hold
  (BCCL, BCCR)
- the Frobenius condition holds
  (Frob)

The precise definition of what a fibration is
is quite technical. I will give it in full here —
it will take several slides — but if you are a
non-specialist you should only pay only attention
to the last two operations: change-of-base functors
and the isomorphism between two changes of base
for the same $g : B \to C$.

Subobjects

Our archetypical fibration will be Cod : CanSub(**Set**) $\to$ **Set**,
where the "codomain functor", Cod, takes each
"canonical subobject" P in **Set**
(where P is an inclusion map $\{\, b \in B \mid P(b) \,\} \hookrightarrow B$)
and returns its codomain, B.

The categorical way to describe "injective maps"
is via the notion of "monic".
The monic arrows of **Set** are exactly the injective maps.
In **Set** *some* injective maps are inclusions.
Let's regard **Set** as category with a (given) distinguished
class of monics — the "inclusions".
A *subobject of* B is a monic $A \rightarrowtail B$
with codomain B.
A *canonical subobject of* B is an inclusion $B' \hookrightarrow B$
with codomain B.

Two subobjects of B, $B' \rightarrowtail B$ and $B'' \rightarrowtail B$
are *isomorphic* when there is an iso $B' \leftrightarrow B''$
making the obvious triangle commute:

$$\begin{array}{ccc} B' & \longleftrightarrow & B'' \\ & \searrow \; \swarrow & \\ & B & \end{array}$$

Each subobject $f : B' \rightarrowtail B$
is isomorphic to a unique canonical subobject of B:

$$\begin{array}{ccl} B' & \longleftrightarrow & \mathrm{Im(f)} = \{\, b \in B \mid f^{-1}(b) \neq \emptyset \,\} \\ & {\scriptstyle f} \searrow & \downarrow \\ & & B \end{array}$$

Subobjects (2)
We will use the following notations for canonical subobjects. In the archetypal case,

explicit, long form:	shorthand (note the '||'!):	

$$
\begin{array}{ccc}
\left(\begin{array}{c} \{\, a \in A \mid P(a) \,\} \\ \downarrow \\ A \end{array}\right) & \{\, a \mid\mid P(a) \,\} & \mathbb{E} = \mathrm{CanSub}(\mathbf{Set}) \\
 & & \downarrow{\scriptstyle p=\mathrm{Cod}} \\
A & A & \mathbb{B} = \mathbf{Set}
\end{array}
$$

and in the generic/abstract case,

$$
\begin{array}{cc}
P & \mathbb{E} \\
 & \downarrow{\scriptstyle p} \\
A & \mathbb{B}
\end{array}
$$

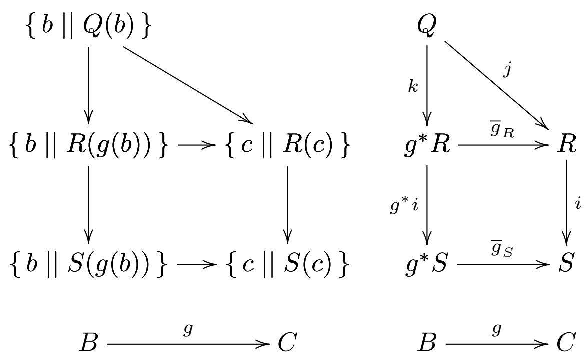
$\mathbb{E}$ is the "entire category",
$\mathbb{B}$ is the "base category",
p is the "projection",
P is a "proposition over A".

Note that instead of drawing a vertical arrow $P \mapsto pP{=}A$ we just draw P over A.
(We will do the same for morphisms, by the way).

Subobjects (3)
If B is an object of $\mathbb{B}$, the *fiber over* B, $\mathbb{E}_B$,
is the subcategory of $\mathbb{E}$ composed of the objects and morphisms
of $\mathbb{E}$ that are taken to B and id_B by the projection p.

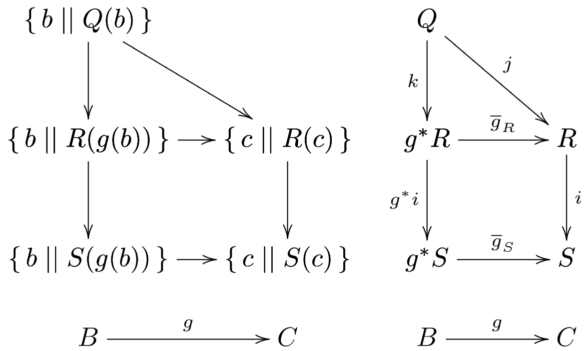
Being "over B" means "belonging to $\mathbb{E}_B$".
A morphism of $\mathbb{E}$ is said to be *vertical* if its image
is an identity in $\mathbb{B}$.
We will have a notion of "horizontal morphisms" too,
but it will be harder to formalize — it will involve *cartesianness*.
Let's start with an example.

Every map $g : B \to C$ in the base category
induces a "change-of-base functor", $g^* : \mathbb{E}_C \to \mathbb{E}_B$,
and a natural transformation $\overline{g} : g^* \to \mathrm{id}_{\mathbb{E}_C}$,
as in the diagram below:

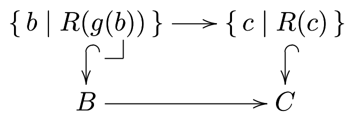


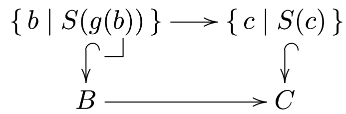
The vertical map $i : R \to S$ exists iff $\{\, c \mid R(c) \,\} \subseteq \{\, c \mid S(c) \,\}$,
i.e., if $\forall c. R(c) \supset S(c)$.
The diagonal map $j : Q \to S$ exists iff its factorization through $\overline{g}_R$, k,
exists (because every map in $\mathbb{E}$ factors as "vertical map followed by
an horizontal map", and we shall see), and:
the vertical map $k : Q \to g^*R$ exists iff $\{\, b \mid Q(b) \,\} \subseteq \{\, b \mid R(g(b)) \,\}$,
i.e., if $\forall b. Q(b) \supset R(g(b))$ — so the vertical/horizontal factorization
gives us a way to interpret diagonal morphisms "logically".

Subobjects (4)
The horizontal maps of the diagram



are pullbacks in **Set**:

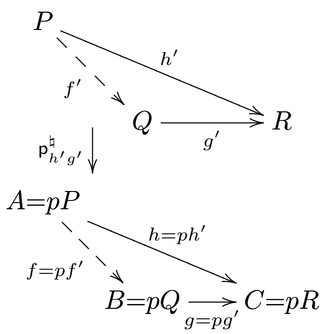




And so what?...

Cartesianness
A *vee* in a category $\mathbb{E}$ is a pair
of morphisms in $\mathbb{E}$, $(h' : P \to R,\ g' : Q \to R)$
with common codomain.
A *completion* for a vee $(h' : P \to R,\ g' : Q \to R)$
is an arrow $f' : P \to Q$ making the triangle commute.
A functor $p : \mathbb{E} \to \mathbb{B}$ takes completions of (h', g')
to completions of (ph', pg'); let's call this induced map $\mathsf{p}^{\natural}_{h'g'}$.

A vee (h', g') in $\mathbb{E}$ has *unique liftings* (for $p : \mathbb{E} \to \mathbb{B}$)
if the map $\mathsf{p}_{h'g'}$ induced by p is a bijection.
A map $g' : Q \to R$ in $\mathbb{E}$ is *cartesian*
if any vee (h', g') with g' as its "lower leg"
has unique liftings.

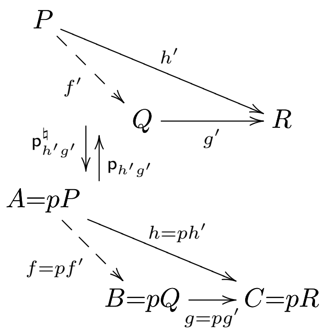


To make things more manageable we will use
some shorthands: $A := pP$, $B := pQ$, $C := pR$,
$f := pf'$, $g := pg'$, $h := ph'$,
Invertibility of $\mathsf{p}^{\natural}_{h'g'}$ means that
for any completion f for the vee (h, g)
there is exactly one completion $f' : P \to Q$ for (h', g')
"over f", i.e., such that $f = pf'$.

Cartesianness (2)
A *proto-vee* in a category $\mathbb{E}$ is a pair
of morphisms in $\mathbb{E}$, $(h' : P \to R,\ g' : Q \to R)$
with common codomain.
A *proto-completion* for a vee $(h' : P \to R,\ g' : Q \to R)$
is an arrow $f' : P \to Q$ ~~making the triangle commute~~.
A functor $p : \mathbb{E} \to \mathbb{B}$ takes proto-completions of (h', g')
to proto-completions of (ph', pg'); let's call this induced map $\mathsf{p}^\natural_{h'g'}$.

Note that this new induced map has a bigger domain...
the old one had to act on all completions,
the new one has to act on all *proto*-completions.

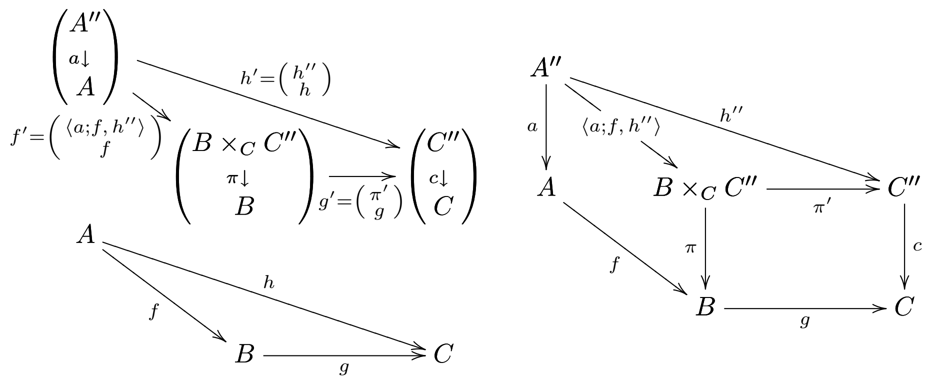
A *vee with proto-unique liftings*
is a triple $(h', g', \mathsf{p}_{h'g'})$,
where $\mathsf{p}_{h'g'}$ is a proto-inverse for $\mathsf{p}^\natural_{h'g'}$.



Proto-cartesianness for an arrow g'
is the assurance that any vee (h', g') with lower leg g'
has proto-unique liftings.
More formally:
a *proto-cartesian morphism* is a pair (g', cart),
where $\mathsf{cart} \equiv (P, h' \mapsto \mathsf{p}_{h'g'})$ is an operation
that produces all the required proto-inverses.

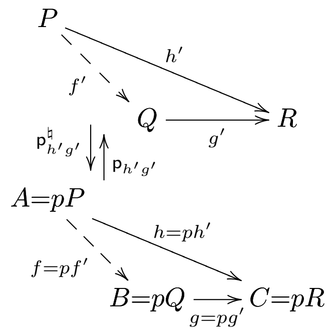
Pullbacks are cartesian
Cartesian maps are not so weird
as the preceding definition suggests...

Fact. When $p = \mathrm{Cod}$, "pullback squares are cartesian".
The core of the proof is the following diagram.



The arrow $g' = \binom{\pi'}{g}$ is a "pullback square".
Fix any $h' = \binom{h''}{h}$ with the same codomain as g'.
For any completion f of the lower vee (h, g),
its lifting is $f' = \binom{\langle a; f, h''\rangle}{f}$.

Cleavages
Here's the abstract view of cartesianness & friends —
formulated in a way that projects well into the syntactical world.
Note that the conditions $pR = C$, $pg' = g$, etc, are always implicit;
the conditions $h = f;g$ and $h' = f';g'$ are implicit
whenever f, g (resp. f', g') are defined
and when we are in the real world;
in the syntactical world they are irrelevant.



Unique liftings for a vee (h', g') is an inverse $\mathsf{p}_{h'g'}$
for the natural operation $\mathsf{p}^\natural_{h'g'}$.
Cartesianness for an arrow g'
is an operation $\mathsf{cart}_{g'} \equiv (P, h' \mapsto \mathsf{p}_{h'g'})$.
A *cartesian lifting* for $g : B \to C$ at R
is a triple $\mathsf{clift}_{gR} \equiv (Q, g', \mathsf{cart}_{g'})$.
A *family of cartesian liftings* for $g : B \to C$
is an operation $\mathsf{clifts}_g \equiv (R \mapsto \mathsf{clift}_{gR})$.
A *cleavage* (for the functor $p : \mathbb{E} \to \mathbb{B}$)
is an operation $\mathsf{cleavage} \equiv (g \mapsto \mathsf{clifts}_g)$.

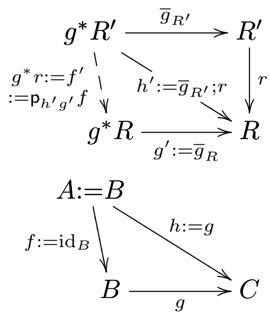
Or all at once:
a cleavage (for the functor $p : \mathbb{E} \to \mathbb{B}$) is an operation
$\mathsf{cleavage} \equiv (g \mapsto (R \mapsto (Q, g', (P, h' \mapsto (f \mapsto f'))))),$
or, more detailedly,
$\mathsf{cleavage} \equiv (B, C, g \mapsto (R \mapsto (Q, g', (A, P, h, h' \mapsto (f \mapsto f'))))).$

A *cloven fibration* is a 4-uple $(\mathbb{B}, \mathbb{E}, p, \mathsf{cleavage})$,
where $p : \mathbb{E} \to \mathbb{B}$ and $\mathsf{cleavage}$ is a cleavage for p.

One further subtlety is needed to make this work in the
syntactical world: as we need a restricted version of
equality to be able to say "P is over A", "f' is over f", etc,
$\mathbb{E}$ should be a "category defined over $\mathbb{B}$".
(Details later!)

Cleavage induces change-of-base
A cheap cloven fibration is just a 4-uple $(\mathbb{B}, \mathbb{E}, p, \mathsf{cleavage})$;
An expensive one also comes with change-of-base functions,
factorization through cartesian morphisms,
rules that say that the composites of cartesians are cartesian,
a family of adjunctions $R \mapsto (p_R \dashv \mathsf{clift_R})$ [← *complete this*],
isos between different changes of base, etc.

Here's how to build the action on morphisms
of a change-of-base functor g^*
($(g^*)_0$ comes from cartesian liftings):



Adjoints to change-of-base
In the archetypal case, $p = \mathrm{Cod} : \mathrm{CanSub}(\mathbf{Set}) \to \mathbf{Set}$,
every change-of-base functor f^* has both a left and a right adjoint.
In some cases these adjoints have simple forms.

Some terminology...
a map $\pi : A \times B \to A$ is a "projection",
and the correspondent $\pi^* : \mathbb{E}_A \to \mathbb{E}_{A\times B}$ is a "weakening functor";
a map $\pi : A \times B \to A \times B \times B$ is a "diagonal",
and the correspondent $\delta^* : \mathbb{E}_{A\times B\times B} \to \mathbb{E}_{A\times B}$ is a "contraction functor".
A π^* "introduces a dummy variable",
a δ^* "collapses two variables".

$$
\begin{array}{ccccccc}
\{\, a,b \mid\mid P(a,b) \,\} & \longmapsto & \{\, a \mid\mid \exists b.P(a,b) \,\} & & P & \longmapsto & \exists_f P \\
\downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow \\
\{\, a,b \mid\mid Q(a) \,\} & \longleftarrow\!\dashv & \{\, a \mid\mid Q(a) \,\} & & f^*Q & \longleftarrow\!\dashv & Q \\
\downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow \\
\{\, a,b \mid\mid R(a,b) \,\} & \longmapsto & \{\, a \mid\mid \forall b.P(a,b) \,\} & & R & \longmapsto & \forall_f R \\
A \times B & \xrightarrow{\pi} & A & & A & \xrightarrow{f} & B
\end{array}
$$

$$
\begin{array}{ccccccccccc}
Pab & \Longrightarrow & \exists b.Pab & & Pab & \Longrightarrow & b{=}b' \land Pabb & & Pa & \Longrightarrow & \exists a.a{=}fb \land Pa \\
\downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow \\
Qa & \Longleftarrow & Qa & & Qabb & \Longleftarrow & Qabb' & & Qfa & \Longleftarrow & Qb \\
\downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow & & \downarrow & \leftrightarrow & \downarrow \\
Rab & \Longrightarrow & \forall b.Rab & & Rab & \Longrightarrow & b{=}b' \supset Rab & & Ra & \Longrightarrow & \forall a.a{=}fb \supset Pa \\
a,b & \longmapsto & a & & a,b & \longmapsto & a,b,b' & & a & \overset{f}{\longmapsto} & b
\end{array}
$$

Rules for the quantifiers
Now our task is to understand why, and in which sense, the adjoints to change-of-base that were just mentioned are "admissible".

In Natural Deduction we have six rules:
$\exists I$, $\exists E$, $\forall I$, $\forall E$, $=I$, $=E$
(a package with all six will be called '$\exists\forall{=}IE$')
and some of them only started to make sense to me (years ago!)
when I saw how to translate them to a sequent-calculus like form,
and how to interpret sequents as inclusions between subsets —
so we will see each of them in its ND form, in the corresponding SC form,
and in a "set-like" form, involving functions, sets and subsets.

An important idea:
any subtree of a derivation in natural deduction
"has semantics", i.e., can be interpreted as an inclusion
between subsets.

The rules are:

$\dfrac{Pa\beta}{\exists b.Pab}\ \exists I$	$\dfrac{\exists b.Pab \qquad \begin{array}{c}[Pab]^1 \quad Qa \\ \vdots \\ Ra\end{array}}{Ra}\ \exists E$
$\dfrac{\begin{array}{c}Qa \\ \vdots \\ Rab\end{array}}{\forall b.Rab}\ \forall I$	$\dfrac{\beta \quad \forall b.Rab}{Ra\beta}\ \forall E$
$\dfrac{}{b{=}b}\ {=}I$	$\dfrac{b{=}b' \quad Pabb}{Pabb'}\ {=}E$

but each of these (apparently) small trees
packs a *huge* amount of information.

Rules for the quantifiers (2)
Here are the translations...

$$\frac{Pa\beta}{\exists b.Pab}\;\exists I$$

$$\frac{(a \mapsto \beta) : A \to B \quad \{\, a, b \mid\mid Pab \,\}}{\{\, a \mid Pa\beta \,\} \hookrightarrow \{\, a \mid \exists b.Pab \,\}}\;\exists I$$

$$\frac{a \vdash \beta \quad \{\, a, b \mid\mid Pab \,\}}{a; Pa\beta \vdash \exists b.Pab}\;\exists I$$

$$\frac{\exists b.Pab \qquad \begin{array}{c}[Pab]^1 \quad Qa \\ \vdots \\ Ra\end{array}}{Ra}\;\exists E$$

$$\frac{\{\, a, b \mid Pab \land Qa \,\} \hookrightarrow \{\, a, b \mid Ra \,\}}{\{\, a \mid (\exists b.Pab) \land Qa \,\} \hookrightarrow \{\, a \mid Ra \,\}}\;\exists E$$

$$\frac{a, b; Pab, Qa \vdash Ra}{a; \exists b.Pab, Qa \vdash Ra}\;\exists E$$

$$\frac{\begin{array}{c}Qa \\ \vdots \\ Rab\end{array}}{\forall b.Rab}\;\forall I$$

$$\frac{\{\, a, b \mid Qa \,\} \hookrightarrow \{\, a, b \mid Rab \,\}}{\{\, a \mid Qa \,\} \hookrightarrow \{\, a \mid \forall b.Rab \,\}}\;\forall I$$

$$\frac{a, b; Qa \vdash Rab}{a; Qa \vdash \forall b.Rab}\;\forall I$$

$$\frac{\beta \quad \forall b.Rab}{Ra\beta}\;\forall E$$

$$\frac{(a \mapsto \beta) : A \to B \quad \{\, a, b \mid\mid Rab \,\}}{\{\, a \mid \forall b.Rab \,\} \hookrightarrow \{\, a \mid Ra\beta \,\}}\;\forall E$$

$$\frac{a \mapsto \beta \quad \{\, a, b \mid\mid Rab \,\}}{a; \forall b.Rab \vdash Ra\beta}\;\forall E$$

$$\frac{}{b{=}b}\;{=}I$$

$$\frac{B}{\{\, b \mid \top \,\} \hookrightarrow \{\, b \mid b{=}b \,\}}\;{=}I$$

$$\frac{B}{b; \top \vdash b{=}b}\;{=}I$$

$$\frac{b{=}b' \quad Pabb}{Pabb'}\;{=}E$$

$$\frac{\{\, a, b, b' \mid\mid Pabb \,\}}{\{\, a, b, b' \mid b{=}b' \land Pabb \,\} \hookrightarrow \{\, a, b, b' \mid Pabb' \,\}}\;{=}E$$

$$\frac{\{\, a, b, b' \mid\mid Pabb' \,\}}{a, b, b'; b{=}b', Pabb \vdash Pabb'}\;{=}E$$

Rules for equality

The three primitive rules for equality in Natural Deduction are:

$$\frac{}{b{=}b}\;{=}I \qquad \frac{b{=}b' \quad Pabb}{Pabb'}\;{=}E \qquad \frac{Pabb \quad \begin{array}{c}[Pabb']^1\\ \vdots\\ Qabb'\end{array}}{Qabb}\;\text{subst};1$$

The four derived rules below deserve proper names.
Expensive ND systems come with them built-in,
but in cheap ND system we have to build them ourselves.

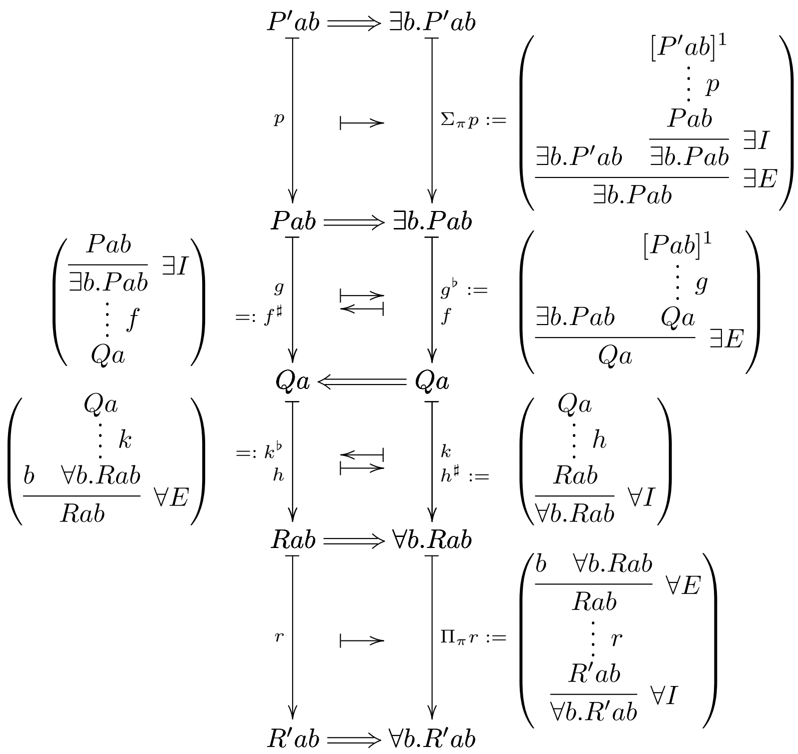
$$\frac{}{b{=}b}\;\text{refl} \quad := \quad \frac{}{b{=}b}\;{=}I$$

$$\frac{b{=}b'}{b'{=}b}\;\text{sym} \quad := \quad \frac{b{=}b' \quad \frac{}{b{=}b}\,{=}I}{b'{=}b}\;{=}E$$

$$\frac{b{=}b' \quad b'{=}b''}{b{=}b''}\;\text{trans} \quad := \quad \frac{b'{=}b'' \quad b{=}b'}{b{=}b''}\;{=}E$$

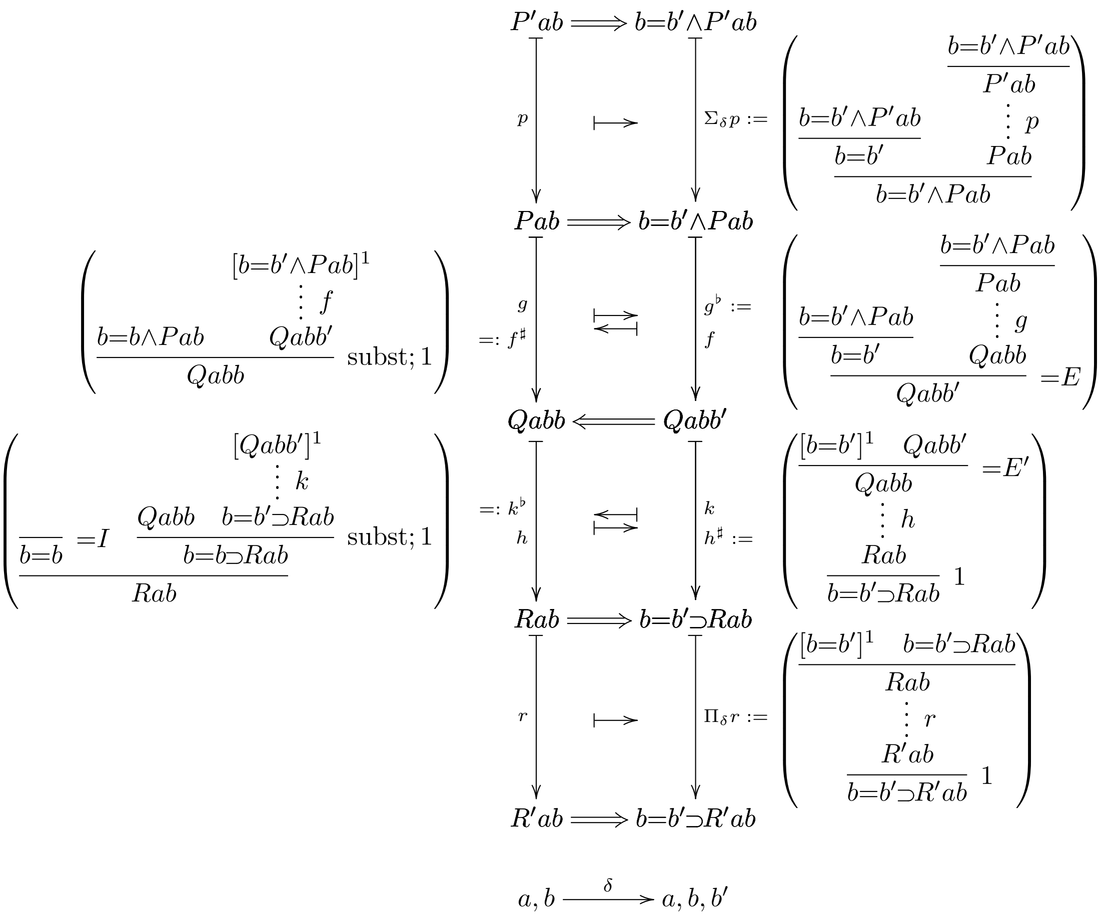
$$\frac{b{=}b' \quad Qabb'}{Qabb}\;{=}E' \quad := \quad \frac{Qabb' \quad \dfrac{b{=}b' \quad \dfrac{[Qabb]^1}{Qabb \supset Qabb}\,1}{Qabb' \supset Qabb}\,{=}E}{Qabb}$$

Adjoints to change-of-base: quantifiers



$$a, b \xrightarrow{\pi} a$$

Adjoints to change-of-base: equality



Adjoints to change-of-base: candidates

Remember that:
$\Delta_B := \langle \mathrm{id}_B, \mathrm{id}_B \rangle : B \to B \times B$,
$\delta_{AB} := \langle \mathrm{id}_{A\times B}, \pi'_{AB} \rangle : A \times B \to (A \times B) \times B$.

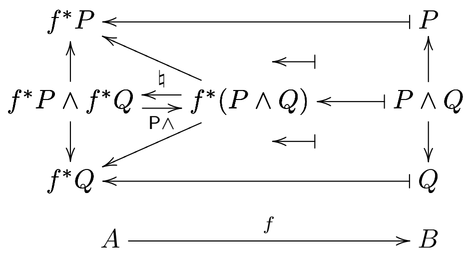
If $f : A \to B$ and $P \in \mathbb{E}_A$ then we can define
$\exists_f P := \exists_{\pi_{BA}}((\mathrm{id}_B \times f)^* \, \mathrm{Eq}_{\Delta_B}\top_B \wedge \pi'^*_{BA}P)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{A \xrightarrow{f} B}{B\times A \to B\times B} \quad \dfrac{B}{\{\, b,b' \mid\mid b{=}b' \,\}}}{\{\, b,a \mid\mid b{=}fa \,\}} \quad \dfrac{\{\, a \mid\mid Pa \,\}}{\{\, b,a \mid\mid Pa \,\}}}{\{\, b,a \mid\mid b{=}fa\wedge Pa \,\}}}{\{\, b \mid\mid \exists a.b{=}fa\wedge Pa \,\}}
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{f}{\mathrm{id}_B\times f} \quad \dfrac{B}{\mathrm{Eq}_{\Delta_B}\top_B}}{(\mathrm{id}_B\times f)^*\,\mathrm{Eq}_{\Delta_B}\top_B} \quad \dfrac{P}{\pi'^*_{BA}P}}{(\mathrm{id}_B\times f)^*\,\mathrm{Eq}_{\Delta_B}\top_B\wedge\pi'^*_{BA}P}}{\exists_{\pi_{BA}}((\mathrm{id}_B\times f)^*\,\mathrm{Eq}_{\Delta_B}\top_B\wedge\pi'^*_{BA}P)}$$

If $P \in \mathbb{E}_{A\times B}$ then we can define
$\mathrm{Eq}_{\delta_{AB}}\, P := (\pi'_{AB} \times \mathrm{id}_B)^* \, \mathrm{Eq}_{\Delta_B}\top_B \wedge \pi^*_{(A\times B)B}P$:

$$\dfrac{\dfrac{\dfrac{B}{\{\, b,b' \mid\mid b{=}b' \,\}}}{\{\, (a,b),b' \mid\mid b{=}b' \,\}} \quad \dfrac{\{\, a,b \mid\mid Pab \,\}}{\{\, (a,b),b' \mid\mid Pab \,\}}}{\{\, (a,b),b' \mid\mid b{=}b'\wedge Pab \,\}}
\qquad
\dfrac{\dfrac{\dfrac{B}{\mathrm{Eq}_{\Delta_B}\top_B}}{(\pi'_{AB}\times\mathrm{id}_B)^*\,\mathrm{Eq}_{\Delta_B}\top_B} \quad \dfrac{P}{\pi^*_{(A\times B)B}P}}{(\pi'_{AB}\times\mathrm{id}_B)^*\,\mathrm{Eq}_{\Delta_B}\top_B\wedge\pi^*_{(A\times B)B}P}$$

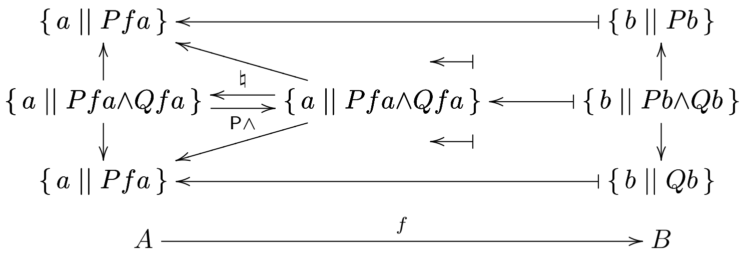
Preservation of 'and'



$$\frac{f \quad P \quad Q}{f^*(P{\wedge}Q) \vdash f^*P{\wedge}f^*Q}\ \mathsf{P}{\wedge}^{\natural} \quad := \quad \cfrac{\cfrac{f \quad \cfrac{P \quad Q}{P{\wedge}Q \vdash P}\ \pi}{f^*(P{\wedge}Q) \vdash f^*P}\ .^* \qquad \cfrac{\cfrac{P \quad Q}{P{\wedge}Q \vdash Q}\ \pi'}{f^*(P{\wedge}Q) \vdash f^*Q}\ .^*}{f^*(P{\wedge}Q) \vdash f^*P{\wedge}f^*Q}\ \langle,\rangle$$

$$\frac{f \quad P \quad Q}{f^*P{\wedge}f^*Q \vdash f^*(P{\wedge}Q)}\ \mathsf{P}{\wedge}$$

Preservation of 'and' (2)
In the archetypal model, CanSub(**Set**),
the arrows $\mathsf{P}\wedge^\natural$ and $\mathsf{P}\wedge$ exist for trivial reasons:
$f^*P \wedge f^*Q = \{\, a \mid\mid Pfa \wedge Qfa \,\} = \{\, a \mid\mid P(f(a)) \wedge Q(f(a)) \,\}$ and
$f^*(P \wedge Q) = \{\, a \mid\mid Pfa \wedge Qfa \,\} = \{\, a \mid\mid P(f(a)) \wedge Q(f(a)) \,\}$
are the *same subobject*.



The same will happen for $\mathsf{P}\top$ and $\mathsf{P}\supset$,
and for *some forms* of Frob, BCCL and BCCR
(the details for Frob, BCCL and BCCR are in [Seely83])

Preservation of 'true'

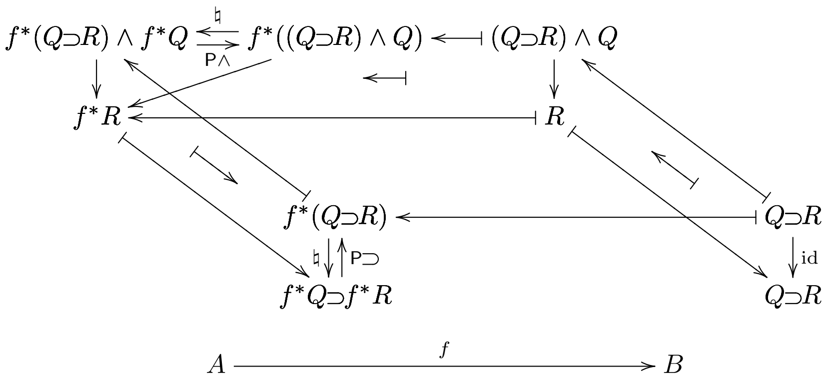
$$\begin{array}{ccc} f^*\top_B & \longleftarrow\!\!\dashv & \top_B \\ {\scriptstyle \mathsf{P}\top^\natural}\big\downarrow\big\uparrow{\scriptstyle \mathsf{P}\top} & & \\ \top_A & & \end{array}$$

$$A \xrightarrow{\;f\;} B$$

$$\frac{f : A \to B}{f^*\top_B \vdash \top_A}\,\mathsf{P}\top^\natural \quad := \quad \dfrac{\dfrac{f \qquad \dfrac{B}{\top_B}\,\top}{f^*\top_B}\,.^*}{f^*\top_B \vdash \top_A}\;!$$

$$\frac{f : A \to B}{\top_A \vdash f^*\top_B}\,\mathsf{P}\top$$

Preservation of 'implies'

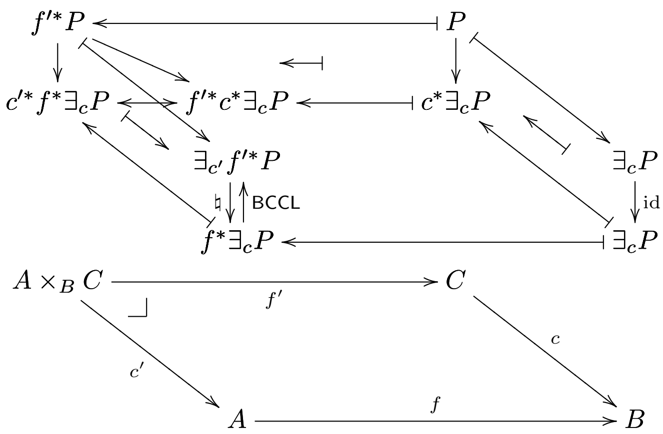


$$\frac{f \quad Q \quad R}{f^*P{\wedge}f^*Q \vdash f^*(Q{\supset}R)}\ \mathsf{P}{\supset}^{\natural} \qquad :=$$

$$\dfrac{\dfrac{\dfrac{f \quad \dfrac{Q \quad R}{Q{\supset}R}\supset \quad Q}{f^*(Q{\supset}R){\wedge}f^*Q \vdash f^*((Q{\supset}R){\wedge}Q)}\ \mathsf{P}\wedge \qquad \dfrac{f \quad \dfrac{\dfrac{\dfrac{Q \quad R}{Q{\supset}R}\supset}{Q{\supset}R \vdash Q{\supset}R}\ \mathrm{id}}{(Q{\supset}R){\wedge}Q \vdash R}\ \mathrm{uncur}}{f^*((Q{\supset}R){\wedge}Q) \vdash f^*R}\ .^*}{f^*(Q{\supset}R){\wedge}f^*Q \vdash f^*R}\ ;}{f^*(Q{\supset}R) \vdash f^*Q{\supset}f^*R}\ \mathrm{cur}$$

$$\frac{f \quad Q \quad R}{f^*(Q{\supset}R) \vdash f^*P{\supset}f^*Q}\ \mathsf{P}{\supset}$$

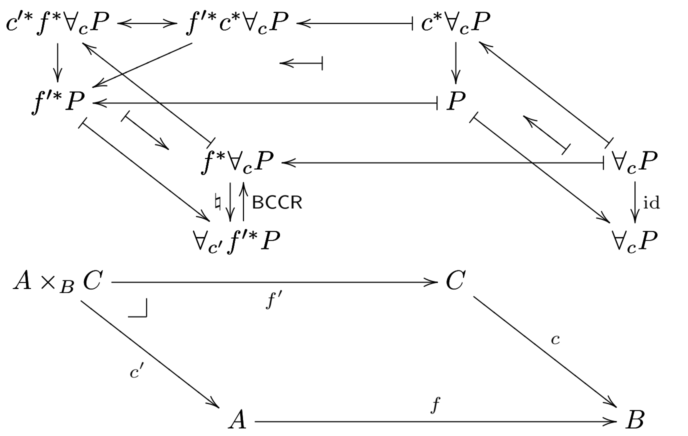
Beck-Chevalley for the left adjoint



$$\frac{c \quad f \quad c' \quad f' \quad P}{\exists_{c'} f'^* P \vdash f^* \exists_c P}\ \mathsf{BCCL}^\natural \quad := \quad \dfrac{\dfrac{\dfrac{f' \quad \dfrac{\dfrac{\dfrac{P \quad c}{\exists_c P}\ \exists_0}{\exists_c P \vdash \exists_c P}\ \mathrm{id}}{P \vdash c^*\exists_c P}\ \exists^\sharp}{f'^*P \vdash f'^* c^* \exists_c P}\ .^* \quad \dfrac{(f';c)=(c';f) \quad \exists_c P}{f'^* c^* \exists_c P \vdash c'^* f^* \exists_c P}}{f'^*P \vdash c'^* f^* \exists_c P}\ ;}{\exists_{c'} f'^* P \vdash f^* \exists_c P}\ \exists^\flat$$

$$\frac{c \quad f \quad c' \quad f' \quad P}{f^* \exists_c P \vdash \exists_{c'} f'^* P}\ \mathsf{BCCL}$$

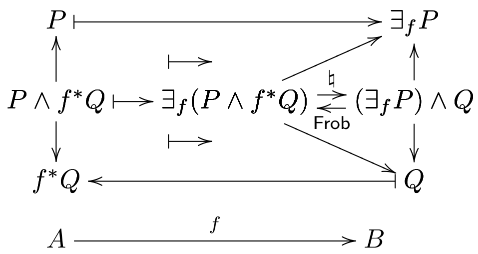
Beck-Chevalley for the right adjoint



$$\frac{c \quad f \quad c' \quad f' \quad P}{f^*\forall_c P \vdash \forall_{c'} f'^* P}\ \mathsf{BCCR}^\natural \quad := \quad \dfrac{\dfrac{\dfrac{(f';c)=(c';f) \quad \forall_c P}{c'^* f^*\forall_c P \vdash f'^* c^*\forall_c P} \quad \dfrac{f' \quad \dfrac{\dfrac{\dfrac{P \quad c}{\forall_c P}\ \forall_0}{\forall_c P \vdash \forall_c P}\ \mathrm{id}}{c^*\forall_c P \vdash P}\ \forall^\flat}{f'^* c^*\forall_c P \vdash f'^* P}\ .^*}{c'^* f^*\forall_c P \vdash f'^* P}\ ;}{f^*\forall_c P \vdash \forall_{c'} f'^* P}\ \forall^\sharp$$

$$\frac{c \quad f \quad c' \quad f' \quad P}{\forall_{c'} f'^* P \vdash f^*\forall_c P}\ \mathsf{BCCR}$$

Frobenius



$$\frac{f \quad P \quad Q}{\exists_f(P\wedge f^*Q) \vdash (\exists_f P)\wedge Q}\ \mathsf{Frob}^\natural \quad := \quad \dfrac{\dfrac{\dfrac{P \quad \dfrac{f \quad Q}{f^*Q}\,.^*}{P\wedge f^*Q \vdash P}\,\pi \quad f}{\exists_f(P\wedge f^*Q) \vdash \exists_f P}\,\exists_f \qquad \dfrac{\dfrac{P \quad \dfrac{f \quad Q}{f^*Q}\,.^*}{P\wedge f^*Q \vdash f^*Q}\,\pi'}{\exists_f(P\wedge f^*Q) \vdash Q}\,{\exists_f}^\flat}{\exists_f(P\wedge f^*Q) \vdash (\exists_f P)\wedge Q}\,\langle,\rangle$$

$$\frac{f \quad P \quad Q}{\exists_f(P\wedge f^*Q) \vdash (\exists_f P)\wedge Q}\ \mathsf{Frob}$$

Cheap hyperdoctrines

A *cheap hyperdoctrine* is a structure like this:

$$\begin{aligned}\mathbb{H} = (&\mathbb{B}, \times, 1, \to, \\ &\mathbb{E}, \land, \top, \supset, \\ &p, \mathsf{cleavage}, \\ &\mathsf{P}\land, \mathsf{P}\top, \mathsf{P}\supset, \\ &\exists_\pi \dashv \pi* \dashv \forall_p, \\ &\mathrm{Eq}_\Delta \dashv \Delta^*, \\ &\mathsf{BCC}\exists_\pi, \mathsf{BCC}\forall_\pi)\end{aligned}$$

Note that Frob does not appear,
and that BCCL and BCCR are stated only for
very special cases — BCC∃, BCC∀, BCC=
(details soon!)

Pimp implies Frob

Fact: we can reconstruct Frob from $\mathsf{P}{\supset}$.
(Note: this is a yellow-belt categorical proof,
but for some reason I find it difficult)
First we need a derived rule:

$$\dfrac{\exists_f(P\land f^*Q)\vdash R}{\exists_f P\land Q\vdash R}\;\mathsf{P}{\supset}' \quad := \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\exists_f(P\land f^*Q)\vdash R}{P\land f^*Q\vdash f^*R}\;\exists^\sharp}{P\vdash f^*Q{\supset}f^*R}\;{\supset}^\sharp}{P\vdash f^*(Q{\supset}R)}\;;\mathsf{P}{\supset}}{\exists_f P\vdash Q{\supset}R}\;\exists^\flat}{\exists_f P\land Q\vdash R}\;{\supset}^\flat$$

where ‘$;\mathsf{P}{\supset}$’ is composition with ‘$\mathsf{P}{\supset}$’, as below.
Note that all the bars in the derivation at the right above
are bijections; the inverse of ‘$;\mathsf{P}{\supset}$’ is ‘$;\mathsf{P}{\supset}^\natural$’.

$$\dfrac{P\vdash f^*Q{\supset}f^*R}{P\vdash f^*(Q{\supset}R)}\;;\mathsf{P}{\supset} \quad := \quad \dfrac{P\vdash f^*Q{\supset}f^*R \qquad \dfrac{f \quad Q \quad R}{f^*Q{\supset}f^*R\vdash f^*(Q{\supset}R)}\;\mathsf{P}{\supset}}{P\vdash f^*(Q{\supset}R)}\;;$$

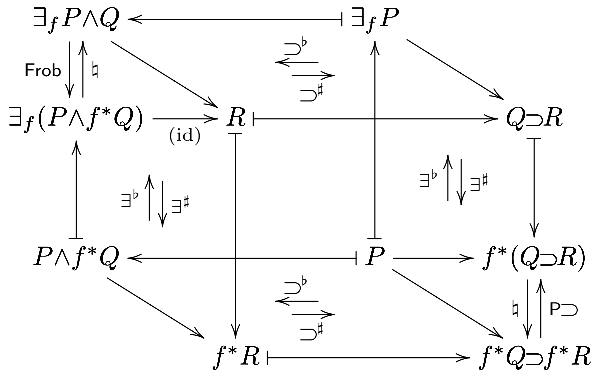
$$\dfrac{P\vdash f^*(Q{\supset}R)}{P\vdash f^*Q{\supset}f^*R}\;;\mathsf{P}{\supset}^\natural \quad := \quad \dfrac{P\vdash f^*(Q{\supset}R) \qquad \dfrac{f \quad Q \quad R}{f^*(Q{\supset}R)\vdash f^*Q{\supset}f^*R}\;\mathsf{P}{\supset}^\natural}{P\vdash f^*Q{\supset}f^*R}\;;$$

Now make $R := \exists_f(P\land f^*Q)$.
We get the tree at the right, below,
that becomes our definition of Frob
as a derived rule (involving $\mathsf{P}{\supset}$).

$$\dfrac{f \quad P \quad Q}{\exists_f P\land Q\vdash \exists_f(P\land f^*Q)}\;\mathsf{Frob} \quad := \quad \dfrac{\dfrac{\dfrac{f \quad P \quad Q}{\exists_f(P\land f^*Q)}}{\exists_f(P\land f^*Q)\vdash\exists_f(P\land f^*Q)}\;\mathrm{id}}{\exists_f P\land Q\vdash \exists_f(P\land f^*Q)}\;\mathsf{P}{\supset}'$$

Pimp implies Frob (2)

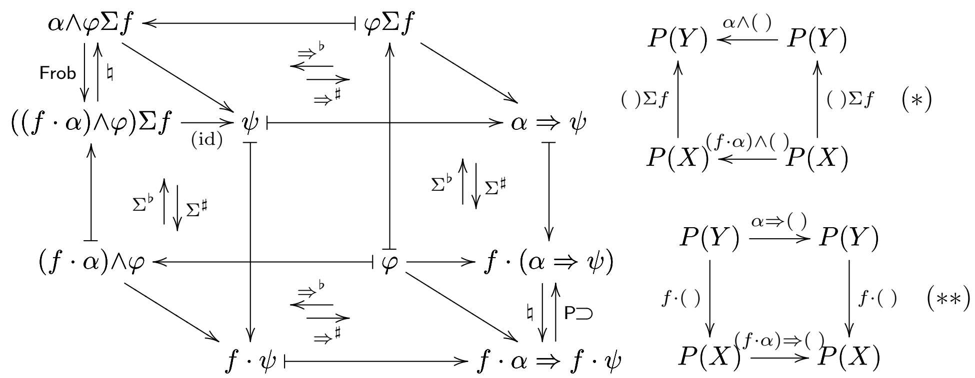
Here is the construction as a diagram.
We start with $R := \exists_f(P \wedge f^*Q)$ and with the arrow marked '(id)',
and we build the arrow $\exists_f P \wedge Q \vdash R = \exists_f(P \wedge f^*Q)$, that is 'Frob'.



Pimp implies Frob (3)
In the notation of [Lawvere70] (p.6), this is stated and proved as:

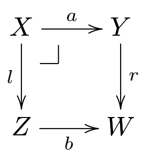
DEFINITION-THEOREM. *In any eed, the following are equivalent:*
(1) Frobenius Reciprocity holds,
(2) For any $f : X \to Y$, α, ψ in $P(Y)$ $\quad f \cdot (\alpha \Rightarrow \psi) \xrightarrow{\approx} f \cdot \alpha \Rightarrow f \cdot \psi$
(3) For any $f : X \to Y$, $\psi \in P(X)$, $\alpha \in P(Y)$ $\quad ((f \cdot \alpha) \wedge \varphi)\Sigma f \xrightarrow{\approx} \alpha \wedge (\varphi \Sigma f)$

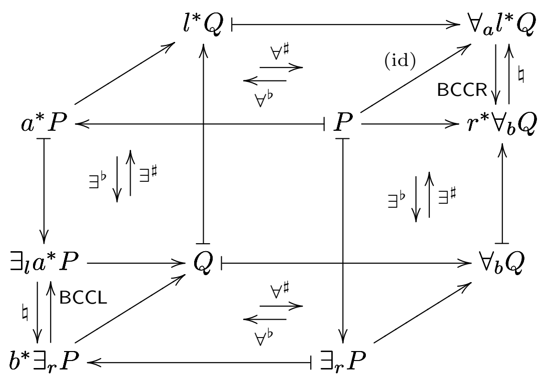
PROOF. The second conditions means that the diagram of functors (**) commutes up to natural equivalence. Hence replacing each functor by its left adjoint also yields a diagram which commutes up to canonical natural equivalence: (*). But the latter is just the third condition. Conversely if the third condition holds, we can replace the functors in the latter diagram by their right adjoints, yielding the second condition.



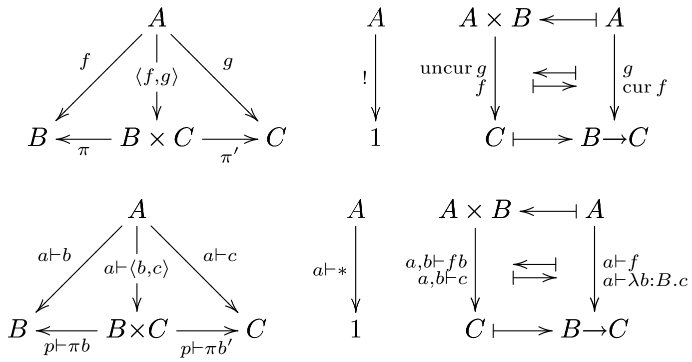
Note that he only draws the squares (*) and (**),
with the categories and the functors;
the cube with objects, morphisms, and '↦'s for functors
is a kind of "internal view" of the categories and functors involved —
and such "internal views" are only very rarely drawn
in the (standard?) Category Theory literature.

BCCL implies BCCR
Also, BCCL implies BCCR (and vice-versa),
by an argument similar to the previous one.
Here are the diagrams; we won't get into the details.





λ-calculus in a hyperdoctrine
As the base category $\mathbb{B}$ of a hyperdoctrine is a CCC we can interpret λ-calculus in it.
In diagrams:



In a sequent-calculus-like form:

$$\frac{B \quad C}{B \times C}\ \times_0 \qquad \frac{a \vdash b \quad a \vdash c}{a \vdash \langle b, c\rangle}\ \langle,\rangle \qquad \frac{B \quad C}{p \vdash \pi p}\ \pi \qquad \frac{B \quad C}{p \vdash \pi' p}\ \pi'$$

$$\frac{}{1}\ 1 \qquad \frac{A}{a \vdash *}\ !$$

$$\frac{B \quad C}{B \to C}\ \to_0 \qquad \frac{B \quad C}{b, f \vdash fb}\ \text{app} \qquad \frac{a, b \vdash c}{a \vdash \lambda b{:}B.c}\ \text{cur}$$

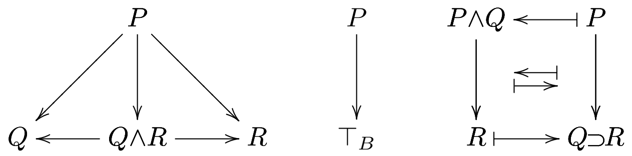
In natural deduction form (in downcased notation):

$$\frac{\begin{matrix} a & a \\ \vdots & \vdots \\ b & c \end{matrix}}{b, c}\ \langle,\rangle \qquad \frac{b, c}{b}\ \pi \qquad \frac{b, c}{c}\ \pi'$$

$$\frac{a}{*}\ !$$

$$\frac{\begin{matrix} a & a \\ \vdots & \vdots \\ b & b \mapsto c \end{matrix}}{c}\ \text{app} \qquad \frac{\begin{matrix} a & [b]^1 \\ & \vdots \\ & c \end{matrix}}{b \mapsto c}\ \lambda;1$$

Propositional calculus in a hyperdoctrine
As each fiber $\mathbb{E}_B$ of a hyperdoctrine is a CCC
we can interpret propositional calculus in it.
In diagrams:



In a sequent-calculus-like form:

$$\frac{Q \quad R}{Q \wedge R}\ \wedge \qquad \frac{P \vdash Q \quad P \vdash R}{P \vdash Q \wedge R}\ \wedge I \qquad \frac{Q \quad R}{Q \wedge R \vdash Q}\ \wedge E_1 \qquad \frac{Q \quad R}{Q \wedge R \vdash R}\ \wedge E_2$$

$$\frac{}{\top_B}\ 1 \qquad \frac{P}{P \vdash \top_B}\ \top I$$

$$\frac{Q \quad R}{Q \supset R}\ \supset \qquad \frac{Q \quad R}{(Q \supset R) \wedge Q \vdash R}\ \mathrm{ev} \qquad \frac{P, Q \vdash R}{P \vdash Q \supset R}\ \supset I$$

In natural deduction form (in downcased notation):

$$\frac{\begin{matrix} P & P \\ \vdots & \vdots \\ Q & R \end{matrix}}{Q \wedge R}\ \wedge I \qquad \frac{Q \wedge R}{Q}\ \wedge E_1 \qquad \frac{Q \wedge R}{R}\ \wedge E_2$$

$$\frac{P}{\top_B}\ \top I$$

$$\frac{\begin{matrix} P & P \\ \vdots & \vdots \\ Q & Q \supset R \end{matrix}}{R}\ \supset E \qquad \frac{\begin{matrix} P & [Q]^1 \\ & \vdots \\ & R \end{matrix}}{Q \supset R}\ \supset I; 1$$

Interpreting '$\forall I$' in a hyperdoctrine

$$\frac{\pi^*Q \vdash R}{Q \vdash \forall_\pi R}\ \forall I
\qquad\qquad
\begin{array}{ccc} \pi^*Q & \longleftarrow\!\!\dashv & Q \\ \downarrow & \longmapsto & \downarrow \\ R & \longmapsto & \forall_\pi R \end{array}$$

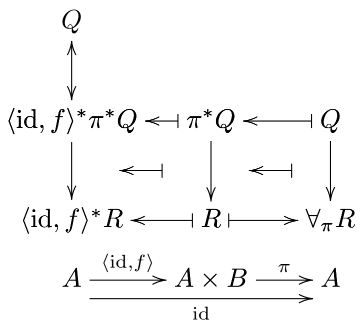
$$A \times B \xrightarrow{\pi} A$$

$$\frac{\begin{array}{c} Qa \\ \vdots \\ Rab \end{array}}{\forall b.Rab}\ \forall I
\qquad
\frac{a,b;Qa \vdash Rab}{a;Qa \vdash \forall b.Rab}\ \forall I
\qquad
\begin{array}{ccc} Qa & \Longleftarrow & Qa \\ \downarrow & \longmapsto & \downarrow \\ Rab & \Longrightarrow & \forall b.Rab \end{array}$$

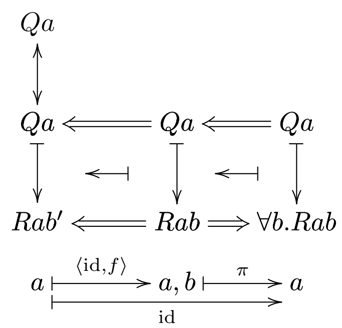
$$a, b \stackrel{\pi}{\longmapsto} a$$

Interpreting '$\forall E$' in a hyperdoctrine

$$\frac{f \quad Q \vdash \forall_\pi R}{Q \vdash \langle \mathrm{id}, f\rangle^* R}\ \forall E$$

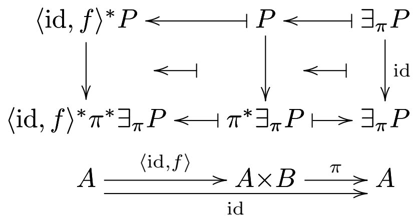


$$\frac{a \vdash b' \quad \begin{matrix} Qa \\ \vdots \\ \forall b.Rab \end{matrix}}{Rab'}\ \forall E \qquad \frac{a \vdash b' \quad a; Qa \vdash \forall b.Rab}{a; Qa \vdash Rab'}\ \forall E$$

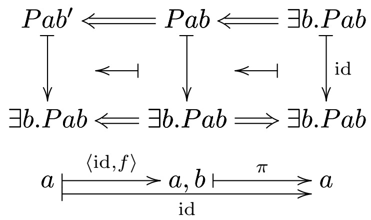


Interpreting '$\exists I$' in a hyperdoctrine

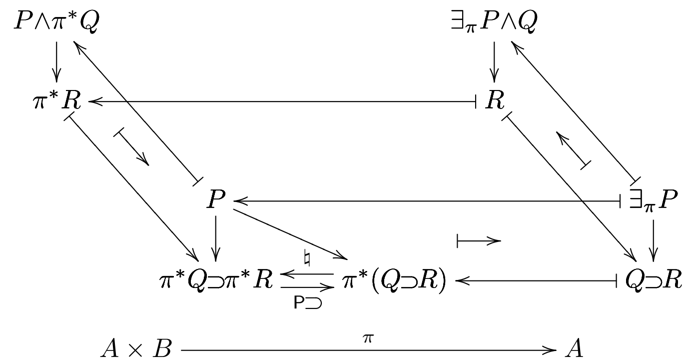
$$\frac{P \wedge \pi^* Q \vdash \pi^* R}{\exists_\pi P \wedge Q \vdash R}\ \exists E$$

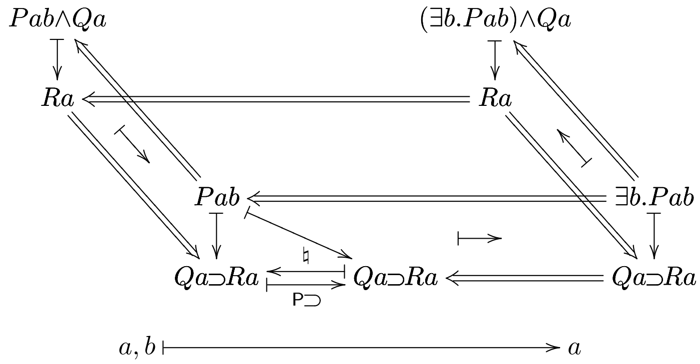


$$\frac{Pab'}{\exists b.Pab}\ \exists I \qquad \frac{a \vdash b' \quad \{\, a, b \mid\mid Pab \,\}}{a; Pab' \vdash \exists b.Pab}\ \exists I$$



Interpreting '$\exists E$' in a hyperdoctrine

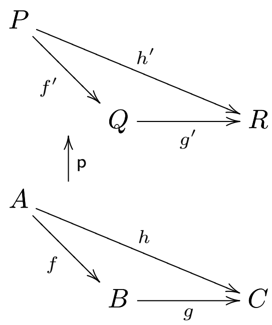




$$\frac{P{\wedge}\pi^*Q \vdash \pi^*R}{\exists_\pi P{\wedge}Q \vdash R}\ \exists E$$

$$\frac{\exists b.Pab \qquad \begin{matrix}[Pab]^1 \quad Qa\\ \vdots\\ Ra\end{matrix}}{Ra}\ \exists E;1 \qquad\qquad \frac{a, b; Pab, Qa \vdash Ra}{a; \exists b.Pab, Qa \vdash Ra}\ \exists E$$

Uppercasing names
When I defined a (proto-)cleavage,
several slides ago, I said that it was:



cleavage $\equiv$
$B, C, R, g \mapsto Q, g', (A, P, f \mapsto f')$

Let's formalize this.

A (proto-)cleavage (for a projection functor $p : \mathbb{E} \to \mathbb{B}$)
is anything that "deserves the name"
$B, C, R, g \mapsto Q, g', (A, P, f \mapsto f')$
that is, a cleavage for $p : \mathbb{E} \to \mathbb{B}$
is any object of a certain type —
where that type is the uppercasing of
$B, C, R, g \mapsto Q, g', (A, P, f \mapsto f')$.

How do we uppercase names as complex as that?
How do we extract the necessary "hints"
from the accompanying diagrams?
Can this always be done unambiguosly?

Uppercasing names (2)
Let's start with a simpler problem.
We have a set A and families of sets

$$\begin{array}{l} a{:}A \vdash B_a \\ a{:}A, b{:}B_a \vdash C_{ab} \\ a{:}A, b{:}B_a, c{:}C_{ab} \vdash D_{abc} \\ a{:}A, b{:}B_a, c{:}C_{ab}, d{:}D_{abc} \vdash E_{abcd} \end{array}$$

and we want to find the type —
written in standard notation for λ-calculus with
dependent types (i.e., with 'Σ's and 'Π's) —
of:

$$a, b, c \vdash d, e.$$

A convention: '$\mathbf{E}[\alpha]$' will mean "the space of 'α's".
We will add lots of '$\mathbf{E}[\ldots]$' entries to our dictionary
to keep track of the types.

Remember:

$$\begin{array}{r} (a, b) : \Sigma a{:}A.B_a \\ (a \mapsto b) : \Pi a{:}A.B_a \end{array}$$

So:

$$\begin{array}{rcl} \mathbf{E}[a] & \equiv & A \\ \mathbf{E}[b] & \equiv & B_a \qquad (\leftarrow \text{which depends on } a) \\ \mathbf{E}[a, b] & \equiv & \Sigma a{:}\mathbf{E}[a].\mathbf{E}[b] \quad \equiv \quad \Sigma a{:}A.B_a \\ \mathbf{E}[a \mapsto b] & \equiv & \Pi a{:}\mathbf{E}[a].\mathbf{E}[b] \quad \equiv \quad \Pi a{:}A.B_a \end{array}$$

Uppercasing names (3)
We want to uppercase

$$a, b, c \mapsto d, e.$$

We start by adding the missing parentheses:

$$(a, (b, c)) \mapsto (d, e)$$

Then:

$$\begin{array}{rcl}
\mathbf{E}[a, (b, c)] & \equiv & \Sigma a{:}\mathbf{E}[a].(\Sigma b{:}\mathbf{E}[b].\mathbf{E}[c]) \\
& \equiv & \Sigma a{:}A.(\Sigma b{:}B_a.C_{ab}) \\
& \equiv & \Sigma a{:}A.\Sigma b{:}B_a.C_{ab} \\
\mathbf{E}[d, e] & \equiv & \Sigma d{:}\mathbf{E}[d].\mathbf{E}[e] \\
& \equiv & \Sigma d{:}D_{abc}.E_{abcd} \\
\mathbf{E}[(a, (b, c)) \mapsto (d, e)] & \equiv & \Pi(a, (b, c)){:}\mathbf{E}[a, (b, c)].\mathbf{E}[d, e] \\
& \equiv & \Pi(a, (b, c)){:}(\Sigma a{:}A.\Sigma b{:}B_a.C_{ab}).(\Sigma d{:}D_{abc}.E_{abcd}) \\
& \equiv & \Pi t{:}(\Sigma a{:}A.\Sigma b{:}B_a.C_{ab}).((\Sigma d{:}D_{abc}.E_{abcd}) \left[\begin{smallmatrix} a:=\pi t, \\ b:=\pi\pi' t, \\ c:=\pi'\pi' t \end{smallmatrix}\right])
\end{array}$$

where the '[...]' on the last line is a substitution box, and:

$$(\Sigma d{:}D_{abc}.E_{abcd}) \left[\begin{smallmatrix} a:=\pi t, \\ b:=\pi\pi' t, \\ c:=\pi'\pi' t \end{smallmatrix}\right] = \Sigma d{:}D_{(\pi t)(\pi\pi' t)(\pi'\pi' t)}.E_{(\pi t)(\pi\pi' t)(\pi'\pi' t)d}$$

when we use then name '$a, (b, c)$' for 't' then it is obvious that:

$$\begin{array}{rcl}
\langle\!\langle a\rangle\!\rangle & := & \pi\langle\!\langle a, (b, c)\rangle\!\rangle \\
\langle\!\langle b, c\rangle\!\rangle & := & \pi'\langle\!\langle a, (b, c)\rangle\!\rangle \\
\langle\!\langle b\rangle\!\rangle & := & \pi\langle\!\langle b, c\rangle\!\rangle \\
\langle\!\langle c\rangle\!\rangle & := & \pi'\langle\!\langle b, c\rangle\!\rangle
\end{array}$$

and so we can avoid choosing a name for the $t \equiv a, (b, c)$,
and we can omit the substitution box...
Long names save the day! 8-)

Uppercasing names (4)
To summarize:

$$
\begin{array}{rcl}
\langle\!\langle \mathbf{E}[a] \rangle\!\rangle & := & A \\
\langle\!\langle \mathbf{E}[b] \rangle\!\rangle & := & B_a \\
\langle\!\langle \mathbf{E}[c] \rangle\!\rangle & := & C_{ab} \\
\langle\!\langle \mathbf{E}[d] \rangle\!\rangle & := & D_{abc} \\
\langle\!\langle \mathbf{E}[e] \rangle\!\rangle & := & E_{abcd} \\
\langle\!\langle a, (b, c) \rangle\!\rangle & := & t \\
\\
\langle\!\langle \mathbf{E}[b, c] \rangle\!\rangle & := & \Sigma b{:}\langle\!\langle \mathbf{E}[b] \rangle\!\rangle.\langle\!\langle \mathbf{E}[c] \rangle\!\rangle \\
\langle\!\langle \mathbf{E}[a, (b, c)] \rangle\!\rangle & := & \Sigma a{:}\langle\!\langle \mathbf{E}[a] \rangle\!\rangle.\langle\!\langle \mathbf{E}[b, c] \rangle\!\rangle \\
\langle\!\langle a \rangle\!\rangle & := & \pi\langle\!\langle a, (b, c) \rangle\!\rangle \\
\langle\!\langle b, c \rangle\!\rangle & := & \pi'\langle\!\langle a, (b, c) \rangle\!\rangle \\
\langle\!\langle b \rangle\!\rangle & := & \pi\langle\!\langle b, c \rangle\!\rangle \\
\langle\!\langle c \rangle\!\rangle & := & \pi'\langle\!\langle b, c \rangle\!\rangle \\
\langle\!\langle \mathbf{E}[d, e] \rangle\!\rangle & := & \Sigma d{:}\langle\!\langle \mathbf{E}[d] \rangle\!\rangle.\langle\!\langle \mathbf{E}[e] \rangle\!\rangle \\
\langle\!\langle \mathbf{E}[(a, (b, c)) \mapsto (d, e)] \rangle\!\rangle & := & \Pi\langle\!\langle a, (b, c) \rangle\!\rangle{:}\langle\!\langle \mathbf{E}[a, (b, c)] \rangle\!\rangle.(\langle\!\langle \mathbf{E}[d, e] \rangle\!\rangle \left[\begin{smallmatrix} a:=\langle\!\langle a \rangle\!\rangle, \\ b:=\langle\!\langle b \rangle\!\rangle, \\ c:=\langle\!\langle c \rangle\!\rangle \end{smallmatrix}\right])
\end{array}
$$

New terminology: a thing like $\langle\!\langle a, b \rangle\!\rangle$ —
a long name within double angle brackets —
is *bracketed long name*, or, for short, a *blong name*.

The dictionary treats blong names as macros.
The dictionary above has two parts.
In the top part we have the "terminals".
In the bottom part we have "(real) macros" (non-terminals).

The definition for $\langle\!\langle a, (b, c) \rangle\!\rangle$ could have been omitted —
the preprocessor would then substitute $\langle\!\langle a, (b, c) \rangle\!\rangle$ by `dnc006`,
which would be a perfectly acceptable name for a variable
(in, say, Coq).

Currying long names
There are natural bijections

$$\begin{array}{ccc}
(a,(b,c)) \mapsto (d,e) & = & f_1 \\
\updownarrow & & \\
a \mapsto ((b,c) \mapsto (d,e)) & = & f_2 \\
\updownarrow & & \\
a \mapsto (b \mapsto (c \mapsto (d,e))) & = & f_3 \\
\updownarrow & & \\
(a,b) \mapsto (c \mapsto (d,e)) & = & f_4 \\
\updownarrow & & \\
((a,b),c) \mapsto (d,e) & = & f_5
\end{array}$$

and usually when we are speaking with humans
we can gloss over the details, and say just:

$$\frac{a \quad b \quad c \quad f}{d, e}\ \text{apps}$$

For f being any of the 'f_i's above.

f_3 — the "totally curried" version of f —
has the simplest type:

$$f_3 : \Pi a{:}A_a.\Pi b{:}B_a.\Pi c{:}C_{ab}.\Sigma d{:}D_{abc}.E_{abcd}$$

and the way to obtain d, e from a, b, c, f_3 is the shortest:

$$\begin{array}{rcl}
(d,e) & = & f_1\,\langle a, \langle b, c\rangle\rangle \\
(d,e) & = & f_2\,a\,\langle b, c\rangle \\
(d,e) & = & f_3\,a\,b\,c \\
(d,e) & = & f_4\,\langle a, b\rangle\,c \\
(d,e) & = & f_5\,\langle\langle a, b\rangle, c\rangle
\end{array}$$

so the "totally curried versions" are usually preferred
in implementations (where we *cannot* gloss over those details)...

Usually in mathematical practice we say $\mathrm{Hom}_{\mathbf{C}}(A,B)$ $(= \mathrm{Hom}_{\mathbf{C}}\langle A,B\rangle)$
but in our implementation of (proto-)CT in type theory
we will use $\mathrm{Hom}_{\mathbf{C}}\,A\,B$.

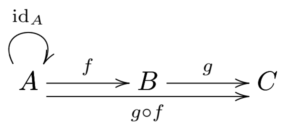
Typing proto-categories
It should be possible to type

$$\mathbf{C} \equiv (\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}})$$

from the equations:

$$\begin{array}{rcl}
A,\ B,\ C & : & \mathbf{C}_0 \\
A \to A & := & \mathrm{Hom}_{\mathbf{C}}\, A\, A \quad : \quad \mathrm{Sets} \\
A \to B & := & \mathrm{Hom}_{\mathbf{C}}\, A\, B \quad : \quad \mathrm{Sets} \\
B \to C & := & \mathrm{Hom}_{\mathbf{C}}\, B\, C \\
A \to C & := & \mathrm{Hom}_{\mathbf{C}}\, A\, C \\
\mathrm{Hom}_{\mathbf{C}} & : & \mathbf{E}[\lambda A.\lambda B.\, \mathrm{Hom}_{\mathbf{C}}\, A\, B] \\
\mathrm{id}_A & := & \mathrm{id}_{\mathbf{C}} A \quad : \quad A \to A \\
\mathrm{id}_{\mathbf{C}} & : & \mathbf{E}[\lambda A.\mathrm{id}_A] \\
g \circ f & := & \circ_{\mathbf{C}}\, A\, B\, C\, g\, f \quad : \quad A \to C \\
\circ_{\mathbf{C}} & : & \mathbf{E}[\lambda A.\lambda B.\lambda C \lambda g.\lambda f.(g \circ f)]
\end{array}$$

And it should be possible
to extract some of them
from this diagram:



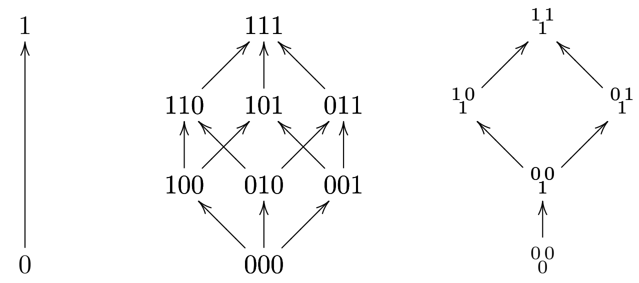
(Note: I'm ignoring all questions of size,
and I'm not saying where $\mathbf{C}_0$ lives).

After that we should have a constant, $\mathbf{E}[\mathbf{C}]$,
that we can use to declare arbitrary (proto-)categories:
a proto-category is an element of $\mathbf{E}[\mathbf{C}]$.

The result should be something like:

$$\begin{array}{rcl}
\mathbf{E}[\mathbf{C}_0] & := & \mathrm{Classes} \\
\mathbf{E}[\mathrm{Hom}_{\mathbf{C}}] & := & \mathbf{C}_0 \to (\mathbf{C}_0 \to \mathrm{Sets}) \\
\mathbf{E}[\mathrm{id}_{\mathbf{C}}] & := & \Pi A{:}\mathbf{C}_0.\, \mathrm{Hom}_{\mathbf{C}}\, A\, A \\
\mathbf{E}[\circ_{\mathbf{C}}] & := & \Pi A{:}\mathbf{C}_0.\Pi B{:}\mathbf{C}_0.\Pi C{:}\mathbf{C}_0. \\
 & & \quad (\mathrm{Hom}_{\mathbf{C}}\, B\, C \to (\mathrm{Hom}_{\mathbf{C}}\, A\, B \to \mathrm{Hom}_{\mathbf{C}}\, A\, C)) \\
\mathbf{E}[\mathbf{C}] & := & \mathbf{E}[\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}}]
\end{array}$$

Introduction to Heyting Algebras
Let's identify 'true' with with 1 and with the singleton set,
and 'false' with 0 and with the empty set.
Now all our truth-values (i.e., 0 and 1) are subsets
of a "top" set, $\top$, and we have an arrow $0 \to 1$,
but there's no arrow $1 \to 0$.
The arrows are "non-decreasing".
In the category $\mathbf{Set}^{\bullet\bullet\bullet}$ we have 8 truth-values,
and in the category $\mathbf{Set}^{\mathbb{V}}$ just five:

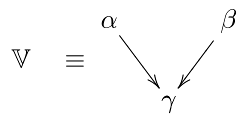


Set $\qquad$ $\mathbf{Set}^{\bullet\bullet\bullet}$ $\qquad$ $\mathbf{Set}^{\mathbb{V}}$

$\mathbb{V}$ is the DAG

$$\mathbb{V} := (W, R) := (\{\,\alpha, \beta, \gamma\,\}, \{\,\alpha \to \gamma, \beta \to \gamma\,\})$$

regarded as a category:



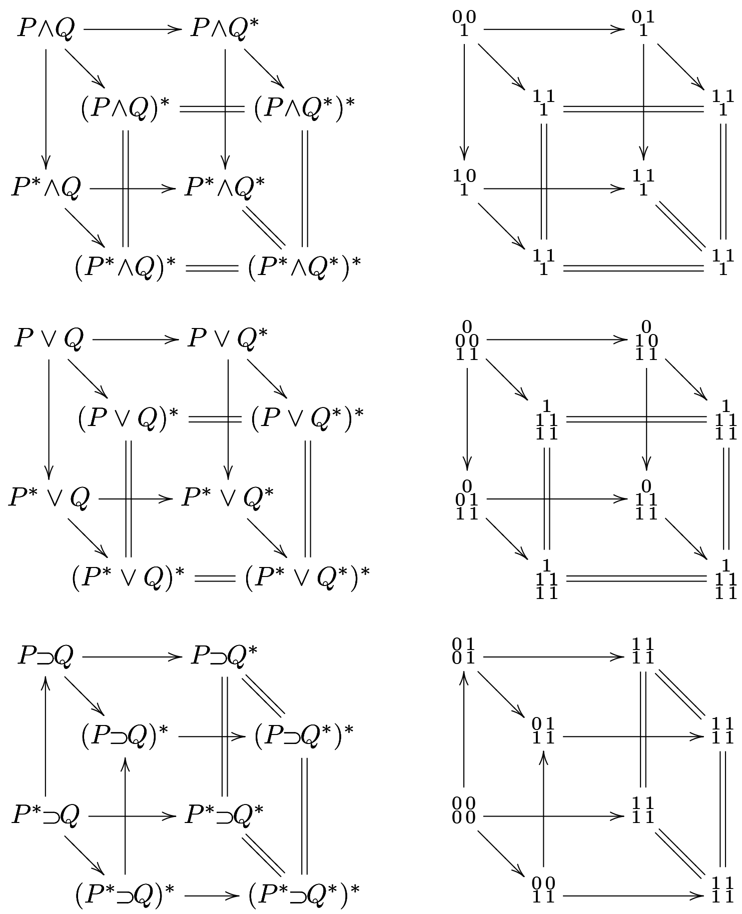
In each category $\mathbf{Set}^{\mathbb{D}}$, where $\mathbb{D}$ is a DAG,
the structure $(\mathrm{Sub}(1_{\mathbb{D}}), \wedge, \top, \supset)$ is a CCC,
and even more: $(\mathrm{Sub}(1_{\mathbb{D}}), \wedge, \top, \supset, \vee, \bot)$ a (bi-)Heyting Algebra.
Its objects behave as *intuitionistic truth-values*...
In $\mathbf{Set}^{\mathbb{V}}$, if we take $P = {}^{0\,0}_{\;1}$ we have
$\neg\neg P = (P \supset \bot) \supset \bot = {}^{1\,1}_{\;1} = \top$, so $P \neq \neg\neg P$.

Introduction to Lawvere-Tierney Topologies

The operation $P \mapsto P^* := \neg\neg P$ on a HA obeys these three rules:

$$\frac{}{\vdash \top^*} \qquad \frac{P \vdash Q}{P^* \vdash Q^*} \qquad \frac{}{P^{**} \vdash P^*}$$

Anything obeying these three rules [in the HA of a topos]
is called a *Lawvere-Tierney topology*.
These three rules are the cheap presentation, of course.
The nicer equivalent expensive presentations for LT-topologies
include how the '*' interacts with $\wedge$, $\top$, $\supset$, $\vee$, $\bot$,



Introduction to Lawvere-Tierney Topologies (2)
...plus how the '*' interacts with $\forall$ and $\exists$,
plus the notions of *-*sheaf* and *sheafification*,
among lots of other things.
With just a little bit more we get *forcing*
and *geometric maps between toposes*.

This all looks very technical, and it is.
However toposes of the form $\mathbf{Set}^{\mathbb{D}}$,
where $\mathbb{D}$ is a small DAG,
seem to be *archetypal*, in a sense...

Map of the world
This is the map of the "obvious" places where my work can be presented.

CT journals — CompSci journals — Alg journals — Logic journals

CT books — CS/TT books — Alg books — Logic books

international conferences (papers) — national conferences (papers)

international conferences (tutorial) — national conferences (tutorial) — proof assistants

math seminars PUC/IMPA/ UFRJ/etc — math seminars at UFF

local CT seminars — local comspci seminars — local algebra seminars — local logic seminars

local categor- ists — local CompScis — local algebra- ists — local logic- ians — local topolog- ists

grad students

iniciação científica

undergrads

There are no (new) theorems in these slides
...because the things that we usually call "theorems"
in Category Theory belong to the real world —
they are a construction *plus something more*.

Take for example the Yoneda Lemma.
It says that given a functor $R : \mathbf{B} \to \mathbf{Set}$
and an object B of $\mathbf{B}$ we have a bijection between the set RB
and the set of natural transformations $C \dot{\to} ((B \to C) \to RC)$.
Here we are working in the syntactical world only
— we *mention* liftings to the real world,
but we don't do any such liftings explicitly,
and so what we get is just the projection of that bijection,
which is a proto-iso between RB
and the set of proto-NTs $C \dot{\to} ((B \to C) \to RC)$.
Usually a "theorem" involving a such construction
would have to either show that it is always a bijection,
or to show a case where it is *not* a bijection.

What we do have here is the definition of the two worlds
(mostly via examples, but whatever...), of the projections,
of the liftings, some ideas of how to work with this
splitting of worlds, and examples.

This is not the kind of work that usually gets published.

Typing proto-adjunctions

$$
\begin{aligned}
&\mathrm{Hom}_{\mathbf{C}} :=A, B \mapsto (A \to B)\\
&\mathrm{id}_{\mathbf{C}} :=A \mapsto \mathrm{id}_A\\
&\circ_{\mathbf{C}} :=A, B, C, f, g \mapsto h\\
&\mathbf{C} :=(\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}})\\
\\
&F_0 :=A \mapsto FA\\
&F_1 :=A, B, f \mapsto Ff\\
&F :=(F_0, F_1)\\
\\
&T_0 :=A \mapsto T_A\\
&T :=(T_0)\\
\\
&\flat_{AB} :=g \mapsto f\\
&\sharp_{AB} :=f \mapsto g\\
&\flat :=A, B \mapsto \flat_{AB}\\
&\sharp :=A, B \mapsto \sharp_{AB}\\
&\eta :=A \mapsto \eta_A\\
&\epsilon :=B \mapsto \epsilon_B\\
&(L \dashv R) :=(\flat, \sharp, \eta, \epsilon)
\end{aligned}
$$

Typing proto-CCCs

$\mathsf{prod}^\natural : =A, h \vdash f, g$
$\mathsf{prod} : =A, f, g \mapsto h$
$B \times C : =(P, \pi, \pi', \mathsf{prod})$
$\times_0 : =B, C \mapsto B \times C$
$\times_1 : =B, C, B', C', \beta, \gamma \mapsto \beta \times \gamma$
$\times : =(\times_0, \times_1)$

$! : =A \mapsto t$
$1 : =(T, !)$

$\mathrm{uncur}_{BC} : =A, g \mapsto f$
$\mathrm{cur}_{BC} : =A, f \mapsto g$
$B{\to}C : =E, \mathrm{ev}_{BC}, \mathrm{cur}_{BC}$
$\to_0 : =B, C \mapsto B{\to}C$

$(\times B)_0 : =A \mapsto A \times B$
$(\times B)_1 : =A', A, \alpha \mapsto \alpha \times B$
$(B{\to})_0 : =C \mapsto (B{\to}C)$
$(B{\to})_1 : =C, C', \gamma \mapsto (B{\to}\gamma)$
$\mathrm{uncur}_B : =A, C, g \mapsto f$
$\mathrm{cur}_B : =A, C, f \mapsto g$
$(\times B) : =((\times B)_0, (\times B)_1)$
$(B{\to}) : =((B{\to})_0, (B{\to})_1)$
$\eta_B : =A \mapsto \mathsf{coev}_{AB}$
$\epsilon_B : =C \mapsto \mathrm{ev}_{BC}$
$((\times B) \dashv (B{\to})) : =(\mathrm{uncur}_B, \mathrm{cur}_B, \eta_B, \epsilon_B)$
$\to : =B \mapsto (B{\to}), ((\times B) \dashv (B{\to}))$

$(\mathbf{C}, \times, 1, \to)$

Typing proto-fibrations

$$
\begin{aligned}
&\mathbb{B} :=(\mathbb{B}_0, \mathrm{Hom}_{\mathbb{B}}, \mathrm{id}_{\mathbb{B}}, \circ_{\mathbb{B}})\\
&\mathbb{E}_0 :=B \mapsto \mathbb{E}_{B0}\\
&\mathrm{Hom}_{\mathbb{E}} :=A, B, f, P, Q \mapsto \mathrm{Hom}_{\mathbb{E}}(P, Q, f)\\
&\mathrm{id}_{\mathbb{E}} :=A, P \mapsto \mathrm{id}_P\\
&\circ_{\mathbb{E}} :=A, B, C, f, g, P, Q, R, f', g' \mapsto h'\\
&\mathbb{E} :=(\mathbb{E}_0, \mathrm{Hom}_{\mathbb{E}}, \mathrm{id}_{\mathbb{E}}, \circ_{\mathbb{E}})
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{ulift}_{h'g'} :=f \mapsto f'\\
&\mathsf{cart}_{g'} :=A, h, P, h' \mapsto \mathsf{ulift}_{h'g'}\\
&\overline{g}_0 :=R \mapsto Q, g', \mathsf{cart}_{g'}\\
&\mathsf{cleavage} :=B, C, g \mapsto \overline{g}_0
\end{aligned}
$$