Motivation

Several years ago I started to have the impression that mathematicians often think using types; several constructions, and demonstrations using those constructions, can be expressed by trees constaing only types — the "skeletons" of the real proofs — and the terms could be added later, in a more or less automatic fashion.

Let's call the problems that can be treated that way "typeful".

The solution of a typeful problem generally goes like this: we discover that there is a natural way to build an object of the desired type from the initial data; we construct that object and check that it has the desired properties (commutes with something, is universal in a certain sense, etc)

We need to define a language for skeletons of proofs and typeful problems before being able to give any examples, but, just to give a first notion of what are the applications:

Calculus and Analysis (??; distant goal; few working examples; slide 16)

Topos theory (?; needs a lot of work)

Basic category theory (looks primising; nice examples).

The Yoneda Lemma is

$$(a; x \Rightarrow x^F) \xrightarrow{\bullet} ((x \xrightarrow{\bullet} ((a \to x) \to x^F)) \leftrightarrow a^F),$$

and, in a certain sense (slide 7), its proof is trivial.

A notation to let us read trees aloud more conveniently

In daily mathematical life *types* are *sets* (i.e.: domains for variables).
Idea: denote a type (= set) by the most natural name that we would choose for a variable ranging over that set. We should be able to recover the name of the set from the name of the variable.
The "typical point" of X is written as x.
Articles: "an x", "the y".

An x is an element of X.
A y is an element of Y.
Now allow some stranger names for variables:
An (a, b) is a pair made of an a and a b ($(a, b) \in A \times B$).
A $c \to d$ is a function that takes each c to a d ($c \to d \in D^C$).

A miniature typeful problem: we have a b and an $(a, b) \to c$, and we need an $a \to c$.

$$\dfrac{\dfrac{[a]^1 \quad b}{(a,b)} \qquad (a,b) \to c}{\dfrac{c}{a \to c}\,1}$$

From an a and a b we [have a natural way to] build an (a, b), ...,
From an a, a b and an $(a, b) \to c$ we build a c,
From a b and an $(a, b) \to c$ we build an $a \to c$.

In most real-world typeful problems each "natural" (or: "interesting") object has a different type; so we'll try to use the names of the types as names for those objects.
Mantra: *objects with related types will have related values.*
If the tree above appeared in our real-world problem then we could read it as:
If we know a and b we know (a, b); ...
If we know b and $(a, b) \to c$ then we know c.

Getting rid of the "uppercase half" of the language
E means "the space of".
$\mathbf{E}_x = X$, $\mathbf{E}_y = Y$,
$\mathbf{E}_{(a,b)} = \mathbf{E}_a \times \mathbf{E}_b$,
$\mathbf{E}_{c\to d} = (\mathbf{E}_d)^{(\mathbf{E}_c)}$
objects with related types are related...
$x \in \mathbf{E}_x$, and so on.

Extending the language to handle categories
We had $\mathbf{E}_x$, $\mathbf{E}_y$ sets,
$x \to y$ a function between those sets.

O will mean "the object of".
$\mathbf{Cat}_\alpha$ is the category where $\mathbf{O}_\alpha$ lives.
$a \to b$ is a morphism from $\mathbf{O}_a$ to $\mathbf{O}_b$ ($\mathbf{Cat}_a = \mathbf{Cat}_b$).

A trick for having more than one value of a given type:
x in $\mathbf{E}_x$, x' in $\mathbf{E}_{x'}$, where $\mathbf{E}_x = \mathbf{E}_{x'}$.
The syntactical rules won't notice that $\mathbf{E}_x$ and $\mathbf{E}_{x'}$ are the same and won't derive $x \leftrightarrow x'$ as a natural arrow.

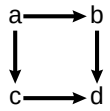
Typically $\mathbf{O}_a, \mathbf{O}_{a'}, \mathbf{O}_{a''}, \ldots$, are objects of the same category, **A**.
Other symbols:
e for "element of": an x is an $\mathbf{e}[\mathbf{E}_x]$
o for "object of": an $\mathbf{O}_a$ is an $\mathbf{o}[\mathbf{A}]$
m for "morphism": an $a' \to a''$ is a $\mathbf{m}[\mathbf{A}]$

Algorithm for attributing meanings to all interesting types
Not yet, so we use a dictionary...
Each problem needs a different dictionary.
Sometimes we will use $[\![\alpha]\!]$ to denote the variable associated to the type α.

$$\left[\!\!\left[\frac{a \quad a \to b}{b}\right]\!\!\right] := [\![a \to b]\!]\,([\![a]\!])$$

$$\begin{aligned}\left[\!\!\left[\dfrac{\dfrac{[a]^1 \quad a \to b}{b} \qquad b \to c}{\dfrac{c}{a \to c}\,1}\right]\!\!\right] &:= \lambda\,[\![a]\!]\,.\left[\!\!\left[\dfrac{\dfrac{a \quad a \to b}{b} \qquad b \to c}{c}\right]\!\!\right] \\ &= \lambda\,[\![a]\!]\,.\,[\![b \to c]\!]\,([\![a \to b]\!]\,([\![a]\!]))\end{aligned}$$

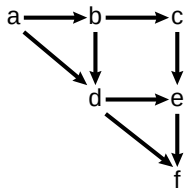
We may impose conditions on the values of variables.
If in our problem we have a commutative diagram



then $[\![b \to d]\!] \circ [\![a \to b]\!] = [\![c \to d]\!] \circ [\![a \to c]\!]$, i.e.,

$$\left[\!\!\left[\frac{a \to b \quad b \to d}{a \to d}\right]\!\!\right] = \left[\!\!\left[\frac{a \to c \quad c \to d}{a \to d}\right]\!\!\right]$$

A tree that builds β from known objects $\alpha_1, \ldots, \alpha_n$ is called a *natural construction* for β (from $\alpha_1, \ldots, \alpha_n$).
If all natural constructions for a β give the same result then we say that β is *well-defined*.
Example: if we know $\mathbf{O}_a, \mathbf{O}_b, \ldots, \mathbf{O}_f, a \to b, \ldots, e \to f$ (i.e., everything that appears explicitly in the diagram below) and we also know that the three cells of the diagram commute then $a \to f$ is well-defined.



Functors and NTs
Our notation for functors describes how the functors act on *names* of objects.

Example: $a \Rightarrow a^F$ is the functor that "adds an F to the superscripts of each object".
It takes $\mathbf{O}[a]$ to $\mathbf{O}[a^F]$,
$\mathbf{O}[a']$ to $\mathbf{O}[a'^F]$,
$a' \to a''$ to $a'^F \to a''^F$,
etc.

Natural transformations: $a \xrightarrow{\bullet} (a^F \to a^G)$ is a NT from $a \Rightarrow a^F$ to $a \Rightarrow a^G$.
It takes $\mathbf{O}[a']$ to $a'^F \to a''^F$.

System NDC, main categorical rules:

$$\frac{\mathbf{O}[a]}{a \to a}\ \mathrm{id} \qquad \frac{a \to b}{\mathbf{O}[a]}\ \mathrm{src} \qquad \frac{a \to b}{\mathbf{O}[b]}\ \mathrm{tgt} \qquad \frac{a \to b \quad b \to c}{a \to c}\ \circ$$

$$\frac{\mathbf{O}[a] \quad a \Rightarrow a^F}{\mathbf{O}[a^F]}\ \Rightarrow\!\mathrm{E}_{\mathbf{O}} \qquad \frac{a' \to a'' \quad a \Rightarrow a^F}{a'^F \to a''^F}\ \Rightarrow\!\mathrm{E}_{\to}$$

$$\frac{\mathbf{O}[a] \quad a \xrightarrow{\bullet} (a^F \to a^G)}{a^F \to a^G}\ \xrightarrow{\bullet}\!\mathrm{E}$$

$$\frac{\dfrac{\overline{[a' \to a'']^1} \quad \ldots}{a'^F \to a''^F}}{a \Rightarrow a^F}\ 1\ \Rightarrow\!\mathrm{I} \qquad \frac{\dfrac{\overline{[\mathbf{O}[a]]^1} \quad \ldots}{a^F \to a^G}}{a \xrightarrow{\bullet} (a^F \to a^G)}\ 1\ \xrightarrow{\bullet}\!\mathrm{I}$$

Example (of $\Rightarrow$I; the second tree is just to explain the types):

$$\frac{\dfrac{\dfrac{\overline{[a' \to a'']^1} \quad a \Rightarrow a^F}{a'^F \to a''^F} \qquad b \Rightarrow b^G}{a'^{FG} \to a''^{FG}}}{a \Rightarrow a^{FG}}\ 1\ \Rightarrow\!\mathrm{I} \qquad \frac{\dfrac{\dfrac{\overline{[\mathbf{m_A}]^1} \quad \mathbf{o}[\mathbf{A} \Rightarrow \mathbf{B}]}{\mathbf{m_B}} \qquad \mathbf{o}[\mathbf{B} \Rightarrow \mathbf{C}]}{\mathbf{m_C}]}}{\mathbf{o}[\mathbf{A} \Rightarrow \mathbf{C}]}\ 1\ \Rightarrow\!\mathrm{I}$$

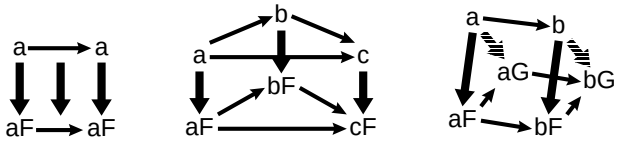
But the rules $\xrightarrow{\bullet}$I and $\Rightarrow$I are wrong, unfortunately.

Protofunctors and protoNTs

The problem is that the rules

$$\dfrac{\dfrac{\overline{[\mathbf{O}[a]]^1} \quad \ldots}{a^F \to a^G}}{a \overset{\bullet}{\to} (a^F \to a^G)}\ 1\ \overset{\bullet}{\to}\mathrm{I} \qquad \dfrac{\dfrac{\overline{[a' \to a'']^1} \quad \ldots}{a'^F \to a''^F}}{a \Rightarrow a^F}\ 1\ \Rightarrow\mathrm{I}$$

are creating operations that behave syntactically like functors and NTs, but that do not need to obey the non-syntactical "naturalness conditions" that functors and NTs obey... namely, that functors respect identities and composition; natural transformations respect the "square condition".



Solution:

A *protofunctor* is defined like a functor, but without the conditions on identity arrows and composition; the image of an $a \to b$ by a protofunctor $a \Rightarrow' a^F$ is any arrow $a^F \to b^F$.

A *protoNT* is defined like a NT but without the square condition: the image of an $\mathbf{O}[a]$ by $a \overset{\bullet}{\to}' (a^F \to a^G)$ is any arrow $a^F \to a^G$.

Notation: $a \Rightarrow' a^F$ is protofunctor, $a \overset{\bullet}{\to}' (a^F \to a^G)$ is a protoNT (note the ticks).

That is, the real rules are $\Rightarrow'\mathrm{I}$ and $\overset{\bullet}{\to}'\mathrm{I}$, and to show that a proto-operation isn't just "proto" we need something else...

$$\dfrac{a \Rightarrow' a^F \quad \text{sth.else}}{a \Rightarrow a^F} \qquad \dfrac{a \overset{\bullet}{\to}' (a^F \to a^G) \quad \text{sth.else}}{a \overset{\bullet}{\to}' (a^F \to a^G)}$$

There are several ways to formalize that "something else" step.

The "something else" step (unprotoizing protooperations)

$$\frac{a \Rightarrow' a^F \quad \text{sth.else}}{a \Rightarrow a^F} \qquad \frac{a \overset{\bullet}{\to}' (a^F \to a^G) \quad \text{sth.else}}{a \overset{\bullet}{\to}' (a^F \to a^G)}$$

First way: the "sth.else" may be a proof that the proto-op isn't just proto; the proof can be done either by checking $F(f \circ g) = F(f) \circ F(g)$ and $F(\mathrm{id}(A)) = \mathrm{id}(F(A))$ (let's discuss just the case of functors for now) or by some metatheorem concerning the form of the tree or the name of the functor.

Second way: we start with a deduction in the system with the rules $\Rightarrow$I and $\overset{\bullet}{\to}$I, forgetting that those rules are wrong. A deduction in the system with $\Rightarrow$I and $\overset{\bullet}{\to}$I may be either a skeleton of a real deduction (if all proto-operations can be shown to be not just proto — checking this is usually trivial but boring), or a "bad skeleton" that can't become a real proof.
Bad skeletons are rare. Why?

Well-behavedness and beyond

Let's call a deduction in the system with $\Rightarrow$I and $\overset{\bullet}{\to}$I *well-behaved* if all the applications of $\Rightarrow$I and $\overset{\bullet}{\to}$I are correct, i.e., can be shown to be not just proto.
Let's call a term β (think as β being the term for the Yoneda lemma, for example) *very well-behaved* if it is well-behaved and all its deductions are equivalent.

Now suppose that for a certain class EWB of terms: 1) checking if a term belongs to that class is trivial; 2) proving that such terms are very well-behaved is trivial (due to a strong metatheorem?), and 3) finding well-behaved deductions for those terms is also trivial (because we've got a good algorithm).
The value of each term β of EWB is "obvious" from the name of the term:
name of $\beta \to$ a deduction for $\beta \to$ value for β
and each arrow is a trivial step.

Some properties of β come straight from the name of β (being a natural transformation, providing a bijection, etc); other properties can be expressed as other terms. If EWB is large enough then these other properties will be obvious too.

Keywords: coherence, parametricity.

Understanding a big term of NDC (Yoneda)
(no formal algorithm yet)

$\mathbf{A} \Rightarrow \mathbf{B}$ is the category of functors from $\mathbf{A}$ to $\mathbf{B}$ (its morphisms are NTs; usual notation: $\mathbf{B}^{\mathbf{A}}$).
An $\mathbf{O}[x \Rightarrow x^F]$ is an $\mathbf{o}[\mathbf{Cat}[x] \Rightarrow \mathbf{Cat}[x^F]]$.
An $\mathbf{O}[a \to b]$ is the set of all '$a \to b$'s (an $\mathbf{o}[\mathbf{Set}]$).
An $\mathbf{O}[x \xrightarrow{\bullet} (x^F \to x^G)]$ is an $\mathbf{o}[\mathbf{Set}]$.

$$
\begin{array}{rl}
(a; x \Rightarrow x^F) \xrightarrow{\bullet} ((x \xrightarrow{\bullet} ((a \to x) \to x^F)) \leftrightarrow a^F) & \\
\mathbf{O}[(a; x \Rightarrow x^F)] & \mathbf{o}[\mathbf{C} \times (\mathbf{C} \Rightarrow \mathbf{Set})] \\
\mathbf{O}[a] & \mathbf{o}[\mathbf{C}] \\
\mathbf{O}[x \Rightarrow x^F] & \mathbf{o}[\mathbf{C} \Rightarrow \mathbf{Set}] \\
\mathbf{O}[x] & \mathbf{o}[\mathbf{C}] \\
\mathbf{O}[x^F] & \mathbf{o}[\mathbf{Set}] \\
(x \xrightarrow{\bullet} ((a \to x) \to x^F)) \leftrightarrow a^F & \mathbf{m}[\mathbf{Set}] \\
\mathbf{E}[x \xrightarrow{\bullet} ((a \to x) \to x^F)] & \mathbf{o}[\mathbf{Set}] \\
x \xrightarrow{\bullet} ((a \to x) \to x^F) & \mathbf{m}[\mathbf{C} \Rightarrow \mathbf{Set}] \\
\mathbf{O}[x] & \mathbf{o}[\mathbf{C}] \\
(a \to x) \to x^F & \mathbf{m}[\mathbf{Set}] \\
\mathbf{E}[a \to x] & \mathbf{o}[\mathbf{Set}] \\
a \to x & \mathbf{m}[\mathbf{C}] \\
\mathbf{O}[a] & \mathbf{o}[\mathbf{C}] \\
\mathbf{O}[x] & \mathbf{o}[\mathbf{C}] \\
\mathbf{E}[x^F] & \mathbf{o}[\mathbf{Set}] \\
\mathbf{E}[a^F] & \mathbf{o}[\mathbf{Set}]
\end{array}
$$

Implicit functors:
$x \Rightarrow (a \to x)$
$(a; x \Rightarrow x^F) \Rightarrow a^F$
$(a; x \Rightarrow x^F) \Rightarrow (x \xrightarrow{\bullet} ((a \to x) \to x^F)$

The types at the right (and some '**E**'s) come from trying to find a sort of a principal type scheme for the term (by hand).

$$\frac{a \leftrightarrow b}{a \to b} \qquad \frac{a \leftrightarrow b}{b \to a} \qquad \frac{a \to b \quad b \to a \quad \text{sth.else}}{a \leftrightarrow b}$$

Translating between our notation and a standard one
CWM, p.61: MacLane states the Yoneda lemma as $\mathrm{Nat}(D(r,-),K) \cong Kr$.
$y : \mathrm{Nat}(D(r,-),K) \to Kr$, the "forward" direction of the bijection:

$$\dfrac{\dfrac{r \in D}{(1_r : r \to r) \in D(r,r)} \quad \dfrac{r \in D \quad [\alpha : D(r,-) \xrightarrow{\bullet} K]^1}{\alpha_r : D(r,r) \to K(r)}}{\dfrac{\alpha_r 1_r \in K(r)}{y = (\alpha \mapsto \alpha_r 1_r) : \mathrm{Nat}(D(r,-),K) \to Kr}\ 1}$$

$$\dfrac{\dfrac{\mathbf{O}[r]}{r \to r} \quad \dfrac{\mathbf{O}[r] \quad [d \xrightarrow{\bullet} ((r \to d) \to d^K)]^1}{(r \to r) \to r^K}}{\dfrac{r^K}{(d \xrightarrow{\bullet} ((r \to d) \to d^K)) \to r^K}\ 1} \qquad \dfrac{\dfrac{\mathbf{o}[D]}{\mathbf{m}[D], \mathbf{e}[\mathbf{E}[r \to r]]} \quad \dfrac{\mathbf{o}[D] \quad [\mathbf{m}[D \Rightarrow \mathbf{Set}]]^1}{\mathbf{m}[\mathbf{Set}]}}{\dfrac{\mathbf{e}[\mathbf{E}[r^K]]}{\mathbf{e}[\mathbf{E}[(d \xrightarrow{\bullet} ((r \to d) \to d^K)) \to r^K]]}\ 1}$$

The other side of the bijection, $y^{-1} : Kr \to \mathrm{Nat}(D(r,-),K)$: (the last discharge step is not shown)

$$\dfrac{\dfrac{\dfrac{\alpha_r 1_r \in K(r) \quad \dfrac{[(f : r \to d) \in D(r,d)]^1 \quad K : D \to \mathbf{Set}}{K(f) : K(r) \to K(d)}}{K(f)(\alpha_r 1_r) \in K(d)}\ 1}{\alpha_d = (f \mapsto K(f)(\alpha_r 1_r)) : D(r,d) \to K(d)}\ 2}{(\alpha : D(r,-) \xrightarrow{\bullet} K) \in \mathrm{Nat}(D(r,-),K)}$$

$$\dfrac{\dfrac{\dfrac{r^K \quad \dfrac{[r \to d]^1 \quad d \Rightarrow d^K}{r^K \to d^K}}{d^K}}{(r \to d) \to d^K}\ 1}{d \xrightarrow{\bullet} ((r \to d) \to d^K)}\ 2 \qquad \dfrac{\dfrac{\dfrac{\mathbf{e}[\mathbf{E}[r^K]] \quad \dfrac{[\mathbf{m}[D]]^1 \quad D \Rightarrow \mathbf{Set}}{\mathbf{m}[\mathbf{Set}]}}{\mathbf{e}[\mathbf{E}[d^K]]}}{\mathbf{m}[\mathbf{Set}]}\ 1}{\mathbf{m}[D \Rightarrow \mathbf{Set}]}\ 2$$

(There are several other steps that are not being shown here)

Terms for other categorical facts

Binary products:
$(x^{\mathrm{op}}; a; b) \xrightarrow{\bullet} ((x \to a, b) \leftrightarrow ((x \to a), (x \to b)))$

Adjunctions: $L \dashv R \iff$
$(a^{\mathrm{op}}; b) \xrightarrow{\bullet} ((a^L \to b) \leftrightarrow (a \to b^R))$

A category has **I**-limits (e.g., pullbacks are $\begin{smallmatrix} & & \bullet \\ & & \downarrow \\ \bullet & \to & \bullet \end{smallmatrix}$-limits) $\iff$
$(x; i \Rightarrow a_i) \xrightarrow{\bullet} ((i \xrightarrow{\bullet} (x \to a_i)) \leftrightarrow (x \to \lim_i a_i))$

Exponential objects (like $\mathbf{E}_{a \to b} = (\mathbf{E}_b)^{(\mathbf{E}_a)}$):
$(x^{\mathrm{op}}; y^{\mathrm{op}}; z) \xrightarrow{\bullet} ((x, y \to z) \leftrightarrow (x \to (y \to z)))$

If we define the subobject functor $x^{\mathrm{op}} \Rightarrow \mathrm{sub}(x)$ then the classifier object (that is $\mathbf{E}_\omega = \{ \top, \dashv \}$ in **Set**) is characterized by:
$x^{\mathrm{op}} \xrightarrow{\bullet} ((x \to \omega) \leftrightarrow \mathrm{sub}(x))$

To prove that a $\mathbf{Set}^{\mathbf{C}}$ is a topos we must check that it has finite limits, exponentials and a classifier; we establish the shorthands
$x := c \Rightarrow x_c$,
$a := c \Rightarrow a_c$,
...,
and we define:
$\mathbf{O}[\lim_i x_i] := c \Rightarrow (\lim_i x_{ic})$,
$\mathbf{O}[(x \to y)] := c \Rightarrow ((c \to \underline{d}) \xrightarrow{\bullet} (x_d \to y_d))$,
$\mathbf{O}_\omega := c \Rightarrow \mathrm{sub}((c \to \underline{d}) \Rightarrow \top)$;
Then we must check that the conditions for being a topos, listed above, are satisfied. Note: $\mathbf{Cat}[(c \to \underline{d})]$ is a slice category (its objects are morphisms of $\mathbf{C}$ with source $\mathbf{O}_c$, a morphism $(c \to \underline{d'}) \to (c \to \underline{d''})$ is a morphism $d' \to d''$ such that the obvious triangle commutes).

$\mathbf{Set}^{\mathbf{C}}$ is a topos: trees
(Warning: these trees use many rules that haven't been discussed earlier).

Here's how to build $\mathbf{O}_{(y\to z)_c} = \mathbf{E}_{(y\to z)_c}$ for a given object $\mathbf{O}_c$, and at the same time show that this construction is contravariant on $\mathbf{O}_y$ and covariant on $\mathbf{O}_z$; Note that to discharge $\mathbf{O}_{(c\to \underline{d})}$ functorially we need to know the category where this object lives, and to build it we need not only $\mathbf{Cat}_c = \mathbf{Cat}_d$ but also $\mathbf{O}_c$ — thus the tree below has a hidden dependency on $\mathbf{O}_c$, which explains why it builds only $(y^{\mathrm{op}}; z) \Rightarrow (y \to z)_c$ and not $(y^{\mathrm{op}}; z) \Rightarrow (c \Rightarrow (y \to z))$, which would be $(y^{\mathrm{op}}; z) \Rightarrow (y \to z)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^1}{\mathbf{O}[d]} \quad [c \Rightarrow y_c]^2}{\mathbf{O}[y_d]}}{(c \to \underline{d}) \Rightarrow y_d}\,1}{\mathbf{O}[(c \to \underline{d}) \Rightarrow y_d]} \quad \dfrac{\dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^3}{\mathbf{O}[d]} \quad [c \Rightarrow z_c]^4}{\mathbf{O}[z_d]}}{(c \to \underline{d}) \Rightarrow z_d}\,3}{\mathbf{O}[(c \to \underline{d}) \Rightarrow z_d]}}{\dfrac{[\mathbf{O}[c \Rightarrow y_c]^{\mathrm{op}}]^4}{\mathbf{O}[(c \to \underline{d}) \Rightarrow y_d]^{\mathrm{op}}}\,2}}{\mathbf{E}[((c \to \underline{d}) \Rightarrow y_d) \to ((c \to \underline{d}) \Rightarrow z_d)]}\,\mathrm{ren}}{\mathbf{E}[(c \to \underline{d}) \xrightarrow{\bullet} (y_d \to z_d)]}}{((c \Rightarrow y_c)^{\mathrm{op}}; c \Rightarrow z_c) \Rightarrow ((c \to \underline{d}) \xrightarrow{\bullet} (y_d \to z_d))}\,4}{(y^{\mathrm{op}}; z) \Rightarrow (y \to z)_c}\,\mathrm{ren}$$

This is functorial on $\mathbf{O}_c$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{O}[(c'' \to \underline{d})]^1}{c'' \to d}}{c' \to d}}{\mathbf{O}[(c' \to \underline{d})]} \quad [(c' \to \underline{d}) \xrightarrow{\bullet} (y_d \to z_d)]^2}{y_d \to z_d}}{(c'' \to \underline{d}) \to (y_d \to z_d)}\,1}{((c' \to \underline{d}) \xrightarrow{\bullet} (y_d \to z_d)) \to ((c'' \to \underline{d}) \to (y_d \to z_d))}\,2}{c \Rightarrow ((c \to \underline{d}) \xrightarrow{\bullet} (y_d \to z_d))}\,3}{c \Rightarrow (y \to z)_c}\,\mathrm{ren}}{\mathbf{O}[y \to z]}\,\mathrm{ren}$$

Uncurrying:

$$\dfrac{\dfrac{[y_c]^1 \qquad \dfrac{\dfrac{\dfrac{[\mathbf{O}[c]]^2}{c \to c}}{\mathbf{O}[c \to c]} \qquad \dfrac{[x_c]^1 \qquad \dfrac{[\mathbf{O}[c]]^2 \qquad \dfrac{x \to (y \to z)}{c \overset{\bullet}{\to} (x_c \to ((c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d)))}\ \text{ren}}{x_c \to ((c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d))}}{(c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d)}}{y_c \to z_c}}{\dfrac{\dfrac{z_c}{\dfrac{\dfrac{}{x_c, y_c \to z_c}\,1}{c \overset{\bullet}{\to} (x_c, y_c \to z_c)}\,2}{x, y \to z}\,\text{ren}}}$$

Currying is trickier, as at first sight $x \to (y \to z)$ "carries more information" than $x, y \to z$.

$$\dfrac{\dfrac{[x_c]^2 \qquad \dfrac{\dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^1}{c \to d} \qquad \dfrac{\mathbf{O}[x]}{a \Rightarrow x_a}}{x_c \to x_d} \qquad \dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^1}{\mathbf{O}[d]} \qquad \dfrac{x, y \to z}{c \overset{\bullet}{\to} (x_c, y_c \to z_c)}\ \text{ren}}{x_d, y_d \to z_d}}{x_d}}{\dfrac{\dfrac{\dfrac{\dfrac{y_d \to z_d}{(c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d)}\,1}{x_c \to ((c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d))}\,2}{c \overset{\bullet}{\to} (x_c \to ((c \to \underline{d}) \overset{\bullet}{\to} (y_d \to z_d)))}\,3}{x \to (y \to z)}\,\text{ren}}$$

The counit (?) of the adjunction, 'ev': $(x, (x \to y)) \to y$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\mathbf{O}[c]]^2}{c \to c}}{\mathbf{O}[(c \to \underline{c})]} \quad \dfrac{}{[(c \to \underline{d}) \xrightarrow{\bullet} (x_d \to y_d)]^1}}{[x_c]^1 \qquad x_c \to y_c}}{y_c}}{x_c, ((c \to \underline{d}) \xrightarrow{\bullet} (x_d \to y_d)) \to y_c}\ 2}{c \xrightarrow{\bullet} (x_c, ((c \to \underline{d}) \xrightarrow{\bullet} (x_d \to y_d)) \to y_c)}\ \text{ren}}{x, (x - y) \to y}$$

The unit (?), $(x, (x \to y)) \to y$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{O}[(c \to \underline{d})]^1}{c \to d} \qquad c \Rightarrow y_c}{[y_c]^2 \qquad y_c \to y_d}}{y_d}}{x_d \to x_d, y_d}\ 1}{(c \to \underline{d}) \xrightarrow{\bullet} (x_d \to x_d, y_d)}\ 2}{y_c \to ((c \to \underline{d}) \xrightarrow{\bullet} (x_d \to x_d, y_d))}\ 3}{c \xrightarrow{\bullet} (y_c \to ((c \to \underline{d}) \xrightarrow{\bullet} (x_d \to x_d, y_d)))}\ \text{ren}}{y \to (x \to x, y)}$$

Building the classifier object and checking that it is a functor:

$$\dfrac{\dfrac{\dfrac{\dfrac{[c' \to c'']^3 \quad \dfrac{[\mathbf{O}[(c'' \to \underline{d})]]^1}{c'' \to d}}{c' \to d}}{\mathbf{O}[(c' \to \underline{d})]} \quad \dfrac{[\mathrm{sub}((c' \to \underline{d}) \Rightarrow \top)]^2}{(c' \to \underline{d}) \Rightarrow \top | Sc'd}\ \mathrm{ren}}{\mathbf{O}[\top | Sc'd]}}{\dfrac{\dfrac{\dfrac{\dfrac{(c'' \to \underline{d}) \Rightarrow \top | Sc'd}{(c'' \to \underline{d}) \Rightarrow \top | Sc''d}\ \mathrm{ren}}{\mathrm{sub}((c'' \to \underline{d}) \Rightarrow \top)}\ \mathrm{ren}}{\mathrm{sub}((c' \to \underline{d}) \Rightarrow \top) \to \mathrm{sub}((c'' \to \underline{d}) \Rightarrow \top)}\ 2}{c \Rightarrow \mathrm{sub}((c \to \underline{d}) \Rightarrow \top)}\ 3}{\dfrac{c \Rightarrow \omega_c}{\mathbf{O}[\omega]}\ \mathrm{ren}}\ 1}\ \mathrm{ren}$$

The 'true' arrow:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{E}[\top]}{(c \to \underline{d}) \Rightarrow \top}\ 1}{\mathrm{sub}((c \to \underline{d}) \Rightarrow \top)}}{\top \hookrightarrow \mathrm{sub}((c \to \underline{d}) \Rightarrow \top)}\ 2}{c \overset{\bullet}{\to} (\top \hookrightarrow \mathrm{sub}((c \to \underline{d}) \Rightarrow \top))}\ 3}{\top \hookrightarrow \omega}\ \mathrm{ren}$$

The classifier rule (that given a monic obtains its characteristic function):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^1}{\mathbf{O}[d]} \quad \dfrac{x' \hookrightarrow x}{c \overset{\bullet}{\to} (x'_c \hookrightarrow x_c)}\ \text{ren}}{x'_d \hookrightarrow x_d} \quad \dfrac{[x_c]^2 \qquad \dfrac{\dfrac{[\mathbf{O}[(c \to \underline{d})]]^1}{c \to d} \quad \dfrac{\dfrac{x' \hookrightarrow x}{\mathbf{O}[x]}}{c \Rightarrow x_c}\ \text{ren}}{x_c \to x_d}}{x_d}}{\top | \exists x' d}}{(c \to \underline{d}) \Rightarrow \top | \exists x' d}\ 1}{\text{sub}((c \to \underline{d}) \Rightarrow \top)}\ \text{ren}}{x_c \to \text{sub}((c \to \underline{d}) \Rightarrow \top)}\ 2}{c \overset{\bullet}{\to} (x_c \to \text{sub}((c \to \underline{d}) \Rightarrow \top))}\ 3}{x \to \omega}\ \text{ren}$$

Pullback with the 'true' arrow:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\mathbf{O}[c]]^2}{c \to c}}{\mathbf{O}[c \to c]} \quad \dfrac{[x_c]^1 \qquad \dfrac{x \to \omega_S}{c \overset{\bullet}{\to} (x_c \to \text{sub}((c \to \underline{d}) \Rightarrow \top))}\ \text{ren}}{x_c \to \text{sub}((c \to \underline{d}) \Rightarrow \top)}}{\text{sub}((c \to \underline{d}) \Rightarrow \top) = (c \to \underline{d}) \overset{\bullet}{\to} (\top | Scd \hookrightarrow \top)}}{\top | Scc \hookrightarrow \top}}{\top \to \omega_S}}{\omega_S}}{x_c \to \omega_S}\ 1}{(x|_S)_c \hookrightarrow x_c}}{c \overset{\bullet}{\to} ((x|_S)_c \hookrightarrow x_c)}\ 2}{x|_S \hookrightarrow x}\ \text{ren}$$

The Inverse Function Theorem and its friends

If $x \to y$ is a smooth function from $\mathbb{R}^n \to \mathbb{R}^n$, with invertible derivative in the point x_0, then the inverse function theorem says that there is a smooth function $y \to x$ that is a local inverse of $x \to y$:

$$\frac{x_0 \quad x \to y \quad \text{sth.else}}{y \to x}$$

Two important corollaries of this can be seen as being just constructions using this rule: the *local form of immersions* ($t \to x, y$ is an immersion, and $t \to x$ is invertible) and the *implicit function theorem* ($x, y \to f$, and we want a function $x \to y$ that keeps f constant $= f_0$, where $f_0 = f(x_0, y_0)$).

$$\dfrac{t \quad \dfrac{\dfrac{t \to x, y}{t \to x}}{x \to t}\,\text{IFT} \qquad t \to x, y}{x \to y} \qquad\qquad \dfrac{\dfrac{x, y \quad x, y \to f}{f} \quad \dfrac{\dfrac{x, y \quad x, y \to f}{f \to (x \to y)}}{x \to y}\,\text{IFT}}{x \to y}$$