Table of Contents

Don't believe any idea here that you can't figure out the details yourself, or you've seen it stated in serious articles written by grown-ups. These notes contain unintentional errors and omissions. Feedback is not only welcome but also desperately needed. Etc, Etc.

Eduardo Ochs
`http://angg.twu.net/`
`http://www.mat.puc-rio.br/~edrx/`
`edrx@inx.com.br`

Note: I'm currently at McGill university, as a visitor — I'm doing a "sandwich PhD" (really!) thanks to a CAPES (`http://www.capes.gov.br/`) grant...

Abstract

(For the FMCS talk)

We will present a logic (system DNC) whose terms represent categories, objects, morphisms, functors, natural transformations, sets, points, and functions, and whose rules of deduction represent certain constructive operations involving those entities. Derivation trees in this system only represent the "T-part" (for "terms" and "types") of the constructions, not the "P-part" ("proofs" and "propositions"): the rules that generate functors and natural transformations do not check that they obey the necessary equations. So, we can see derivations in this system either as constructions happening in a "syntactical world", that should be related to the "real world" in some way (maybe through metatheorems that are yet to be found), or as being just "skeletons" of the real constructions, with the P-parts having been omitted for briefness.

Even though derivations in DNC tell only half of the story, they still have a certain charm: DNC terms represent "types", but a tree represents a construction of a lambda-calculus term; there's a Curry-Howard isomorphism going on, and a tree can be a visual help for understanding how the lambda-calculus term works — how the data flows inside a given categorical construction. Also, if we are looking for a categorical entity of a certain "type" we can try to express it as a DNC term, and then look for a DNC "deduction" having it as its "conclusion"; the deduction will give us a T-part, and we will still have to go back to the standard language to supply a P-part, but at least the search has been broken in two parts...

The way to formalize DNC, and to provide a translation between terms in its "logic" and the usual notations for Category Theory, is based on the following idea. Take a derivation tree D in the Calculus of Constructions, and erase all the contexts and all the typings that appear in it; also erase all the deduction steps that now look redundant. Call the new tree D'. If the original derivation, D, obeys some mild conditions, then it is possible to reconstruct it — modulo exchanges and unessential weakenings in the contexts — from D', that is much shorter. The algorithm that does the reconstruction generates as a by-product a "dictionary" that tells the type and the "minimal context" for each term that appears in D'; by extending the language that the dictionary can deal with we get a way to translate DNC terms and trees — and also, curiously, with a few tricks more, and with some minimal information to "bootstrap" the dictionary, categorical diagrams written in a DNC-like language.

Motivation

A language for "skeletons of proofs" that would...

- let me write down just the essential — an (imaginary) computer would fill out the missing details;
- allow me say things like "the natural entity of type/name $a, (a \mapsto b) \mapsto b$" (ev);
- have a diagrammatic version,
- and a linear version (for derivation trees and Curry-Howard);
- easy to translate to the usual languages;
- be able to represent categorical concepts;
- allow for the "right amount" of ambiguity;
- read aloud as something that's happening in the simplest models — e.g., in **Set** instead of in an arbitrary topos;
- be concise and clear enough — because I have a very bad memory and I tended to change my notations madly, so I needed short diagrams that would "say everything", without the need for pages of definitions...

Sketch of the solution

A notation that looks like an ambiguous λ-calculus (I call it the "generic point" notation);

add some symbols to GP to let it represent functors, natural transformations, objects, categories, etc;

a dictionary will translate GP terms into something precise (in the Calculus of Constructions)

(diagrams and derivation trees are recyclable! Just change the dictionary...)

modify the Calculus of Constructions to get something that looks more like natural deduction — i.e., that has discharges instead of "$\ldots \vdash \ldots$"s — to get rid of some redundancy

The system of natural deduction for categories ("system DNC") is essentially this modified CC, with the GP notation with categorical extensions, and a dictionary.

DNC derivations can either represent just the "T-part" (terms and types) of categorical constructions, and omit the "P-part" (propositions and proofs), or they can be complete.

The “generic point” notation

What would be the natural name for a variable in...

A	$\dashrightarrow$	a	$A = \mathbf{E}[a]$
B	$\dashrightarrow$	b	$B = \mathbf{E}[b]$
$A \times B$	$\dashrightarrow$	a, b	$A \times B = \mathbf{E}[a, b] = \mathbf{E}[a] \times \mathbf{E}[b]$
B^A	$\dashrightarrow$	$a \mapsto b$	$B^A = \mathbf{E}[a \mapsto b] = \mathbf{E}[b]^{\mathbf{E}[a]}$

We will allow strange names for variables and terms
and we will have a dictionary to translate these names
into something precise (in the Calculus of Constructions).

‘**E**’ is pronounced as “the space of”.

Functors and NTs in this notation

Choose sets A and B, and a function $B \xrightarrow{g} A$.

In GP/DNC notation we write $\mathrm{Hom}(A,-)$ as $x \Rightarrow (a \mapsto x)$.

Note that from the name "$x \Rightarrow (a \mapsto x)$" we can extract both the action on objects, $\mathbf{E}[x] \mapsto \mathbf{E}[a \mapsto x]$, and the action on morphisms, $(x \mapsto y) \mapsto ((a \mapsto x) \mapsto (a \mapsto y))$.

$$\begin{tikzcd}
x \arrow[r, Rightarrow] \arrow[d, mapsto] \arrow[rd, phantom, "\longmapsto"] & (a \mapsto x) \arrow[d, mapsto] \\
y \arrow[r, Rightarrow] & (a \mapsto y)
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[r, mapsto] \arrow[d, "f"'] \arrow[rd, phantom, "\longmapsto"] & \mathrm{Hom}(A,X) \arrow[d, "\mathrm{Hom}(A,f)"] \\
Y \arrow[r, mapsto] & \mathrm{Hom}(A,Y)
\end{tikzcd}$$

$x \overset{\bullet}{\to} ((b \mapsto x) \mapsto (a \mapsto x))$ is a natural transformation going from $x \Rightarrow (b \mapsto x)$ to $x \Rightarrow (a \mapsto x)$.

It works as $\mathbf{E}[x] \mapsto ((b \mapsto x) \mapsto (a \mapsto x))$, but it also obeys a naturality condition...

$$\begin{tikzcd}
& x \arrow[ld, Rightarrow] \arrow[rd, Rightarrow] \arrow[d, "\bullet"] & \\
(b \mapsto x) \arrow[rr, mapsto] & {} & (a \mapsto x)
\end{tikzcd}
\qquad
\begin{tikzcd}
& X \arrow[ld, mapsto, "\mathrm{Hom}(B,-)"'] \arrow[rd, mapsto, "\mathrm{Hom}(A,-)"] \arrow[d, mapsto] & \\
\mathrm{Hom}(B,X) \arrow[rr] & {} & \mathrm{Hom}(A,X)
\end{tikzcd}$$

The proof of the Yoneda Lemma in DNC

$$\dfrac{\dfrac{\mathbf{O}[a]\quad[\mathbf{O}[x]]^1}{\mathbf{E}[a\mapsto x]}\;\mathrm{Hom}_{\mathbf{C}}}{\mathbf{O}[x]\mapsto\mathbf{E}[a\mapsto x]}\;1 \qquad \dfrac{\dfrac{\dfrac{[a\mapsto x']^1\quad[x'\mapsto x'']^2}{a\mapsto x''}}{(a\mapsto x')\mapsto(a\mapsto x'')}\;1}{x\Rightarrow(a\mapsto x)}\;(\Rightarrow I);2$$

$$\dfrac{\dfrac{\dfrac{\mathbf{O}[a]}{a\mapsto a}\quad\dfrac{\mathbf{O}[a]\qquad\dfrac{\dfrac{\dfrac{\dfrac{\underline{\underline{\mathbf{O}[a]}}}{x\Rightarrow(a\mapsto x)}}{\mathbf{O}[x\Rightarrow(a\mapsto x)]}\;\mathrm{ren}\quad\mathbf{O}[x\Rightarrow x^F]}{\mathbf{E}[x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F)]}\;\mathrm{Hom}_{\mathbf{Set^C}}}{x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F)}\;[s]^1}{(a\mapsto a)\mapsto a^F}}{a^F}}{(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F))\mapsto a^F}\;1$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{O}_a\quad\dfrac{\mathbf{O}[x\Rightarrow x^F]}{x\Rightarrow x^F}\;\mathrm{ren}}{\mathbf{E}[a^F]}}{a^F}\;[s]^3\qquad\dfrac{\dfrac{\dfrac{\mathbf{O}_a\quad[\mathbf{O}_x]^2}{\mathbf{E}[a\mapsto x]}}{a\mapsto x}\;[s]^1\quad\dfrac{\mathbf{O}[x\Rightarrow x^F]}{x\Rightarrow x^F}\;\mathrm{ren}}{a^F\mapsto x^F}}{x^F}}{(a\mapsto x)\mapsto x^F}\;1}{x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F)}\;2}{a^F\mapsto(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F))}\;3$$

$$\dfrac{\dfrac{\dfrac{\underline{\underline{[\mathbf{O}_a]^1\quad[\mathbf{O}[x\Rightarrow x^F]]^1}}}{a^F\mapsto(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F))}\quad\dfrac{\underline{\underline{[\mathbf{O}_a]^1\quad[\mathbf{O}[x\Rightarrow x^F]]^1}}}{(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F))\mapsto a^F}}{a^F\leftrightarrow(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F))}}{(a;x\Rightarrow x^F)\overset{\bullet}{\to}(a^F\leftrightarrow(x\overset{\bullet}{\to}((a\mapsto x)\mapsto x^F)))}\;1$$

The proof of the Yoneda Lemma in DNC (2)
...and if we want to prove that the functors implicit in the outer '$\overset{\bullet}{\to}$' of $(a; x \Rightarrow x^F) \overset{\bullet}{\to} (a^F \leftrightarrow (x \overset{\bullet}{\to} ((a \to x) \to x^F)))$ are really (proto!) functors, we need this:

$$\begin{array}{c} \begin{array}{c} [a' \mapsto a'']^1 \qquad \begin{array}{c} [(x \Rightarrow x^{F'}) \mapsto (x \Rightarrow x^{F''})]^1 \\ \hline x \overset{\bullet}{\to} (x^{F'} \mapsto x^{F''}) \end{array} \text{ren} \\ \hline\hline {a'}^{F'} \mapsto {a''}^{F''} \end{array} \\ \hline (a; x \Rightarrow x^F) \Rightarrow a^F \end{array} \; 1$$

$$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} [\mathbf{O}_a]^1 \\ \hline\hline x \Rightarrow (a \mapsto x) \end{array} \\ \hline \mathbf{O}[x \Rightarrow (a \mapsto x)] \end{array} \text{ren} \qquad [\mathbf{O}[x \Rightarrow x^F]]^1 \\ \hline \mathbf{E}[x \overset{\bullet}{\to} ((a \mapsto x) \mapsto x^F)] \end{array} \text{Hom}_{\mathbf{Set}^{\mathbf{C}}} \\ \hline \mathbf{O}[a; x \Rightarrow x^F] \mapsto \mathbf{E}[x \overset{\bullet}{\to} ((a \mapsto x) \mapsto x^F)] \end{array} \; 1$$

$$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} [a' \mapsto a'']^4 \quad [a'' \mapsto x]^1 \\ \hline a' \mapsto x \end{array} \qquad \begin{array}{c} [\mathbf{O}_x]^2 \quad [x \overset{\bullet}{\to} ((a' \mapsto x) \mapsto x^{F'})]^3 \\ \hline (a' \mapsto x) \mapsto x^{F'} \end{array} \\ \hline x^{F'} \end{array} \qquad \begin{array}{c} [\mathbf{O}_x]^2 \quad [x \overset{\bullet}{\to} (x^{F'} \mapsto x^{F''})]^4 \\ \hline x^{F'} \mapsto x^{F''} \end{array} \\ \hline x^{F''} \end{array} \\ \hline (a'' \mapsto x) \mapsto x^{F''} \end{array} \; 1 \\ \hline x \overset{\bullet}{\to} ((a'' \mapsto x) \mapsto x^{F''}) \end{array} \; (\overset{\bullet}{\to} I); 2 \\ \hline (x \overset{\bullet}{\to} ((a' \mapsto x) \mapsto x^{F'})) \mapsto (x \overset{\bullet}{\to} ((a'' \mapsto x) \mapsto x^{F''})) \end{array} \; 3 \\ \hline (a; x \Rightarrow x^F) \Rightarrow (x \overset{\bullet}{\to} ((a \mapsto x) \mapsto x^F)) \end{array} \; (\Rightarrow I); 4$$

Adjunctions as DNC diagrams
$L \dashv R$ is $(a;b) \overset{\bullet}{\to} ((a^L \mapsto b) \leftrightarrow (a \mapsto b^R))$
$(-) \times B \dashv (-)^B$ is $(a;c) \overset{\bullet}{\to} ((a, b \mapsto c) \leftrightarrow (a \mapsto (b \mapsto c)))$

$$\begin{tikzcd}
a^L \arrow[d, mapsto] & a \arrow[l, Rightarrow, "L"'] \arrow[d, mapsto] & & a, b \arrow[d, mapsto] & a \arrow[l, Rightarrow, "(-)\times B"'] \arrow[d, mapsto] \\
b \arrow[r, Rightarrow, "R"'] & b^R & & c \arrow[r, Rightarrow, "(-)^B"'] & b \mapsto c
\end{tikzcd}$$

The functor from $\mathrm{Sub}(A)$ to $\mathrm{Sub}(A \times B)$ (A, B sets, for example) has a left adjoint $\exists$ and a right adjoin $\forall$.

The weakening functor induced by $\Gamma, a \mapsto \Gamma$ in a "good" codomain fibration has a left adjoint Σ and a right adjoint Π.

$$\begin{tikzcd}
a|_{\exists b.P(a,b)} \arrow[d, mapsto] & (a,b)|_{P(a,b)} \arrow[l, Rightarrow, "\exists"'] \arrow[d, mapsto] & & \begin{pmatrix}\Gamma, a, p \\ \downarrow \\ \Gamma\end{pmatrix} \arrow[d, mapsto] & \begin{pmatrix}\Gamma, a, p \\ \downarrow \\ \Gamma, a\end{pmatrix} \arrow[l, Rightarrow, "\Sigma"'] \arrow[d, mapsto] \\
a|_{Q(a)} \arrow[r, Rightarrow] \arrow[d, mapsto] & (a,b)|_{Q(a)} \arrow[d, mapsto] & & \begin{pmatrix}\Gamma, q \\ \downarrow \\ \Gamma\end{pmatrix} \arrow[r, Rightarrow] \arrow[d, mapsto] & \begin{pmatrix}\Gamma, a, q \\ \downarrow \\ \Gamma, a\end{pmatrix} \arrow[d, mapsto] \\
a|_{\forall b.R(a,b)} & (a,b)|_{R(a,b)} \arrow[l, Rightarrow, "\forall"] & & \begin{pmatrix}\Gamma, (a \mapsto r) \\ \downarrow \\ \Gamma\end{pmatrix} & \begin{pmatrix}\Gamma, a, r \\ \downarrow \\ \Gamma, a\end{pmatrix} \arrow[l, Rightarrow, "\Pi"]
\end{tikzcd}$$

Introduction to the Calculus of Constructions (1)

CC is a particular case of a *pure type system*.
A PTS has a set of *sorts* (for CC, $\{*, \Box\}$),
a set of *axioms* involving sorts (for CC, $\{\vdash *{:}\Box\}$),
and a set of pairs of sorts (for CC, $\{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}$)
that specify which variations of the derivation rules are valid.
We also need a countable set of *variables* —
e.g. $\{a, b, c, \ldots, A, B, C, \ldots\}$.
From that we define *pre-terms*, *pre-judgements* and (valid) derivations.
Terms and judgements are pre-terms and pre-judgements that
can appear in valid derivations.

Pre-terms and pre-judgements

The set of pre-terms is the least set closed under the following rules:

- sorts are pre-terms
- variables are pre-terms

If v is a variable, f, t, t' pre-terms,

- ft is a pre-term (application)
- $\lambda v{:}t.t'$ is a pre-term (abstraction, function)
- $\Pi v{:}t.t'$ is a pre-term (function space)

Examples of (pre-)terms:
$\lambda A{:}*.\lambda a{:}A.a$ (polymorphic identity)
$\lambda I{:}*.\lambda A{:}(\Pi i{:}I.*).(\Pi i{:}I.Ai)$

A pre-judgement is something like
$x_1{:}A_1, \ldots, x_n{:}A_n \vdash t{:}t'$
where $x_1, \ldots, x_n$ are all distinct
and $A_1, \ldots, A_n, t, t'$ are pre-terms.
n can be 0 (“empty context”).

Naive semantics: λ and Π
The λ of CC is like the usual λ of λ-calculus, but with a restriction on the type of the input.
$(\lambda a{:}A.b)a'$ will only be a (valid) term if $a'{:}A$.

The Π forms function spaces:
If $b : B$ then $(\lambda a{:}A.b){:}(\Pi a{:}A.B)$.

Dependent types: in $\lambda a{:}A.b$ and $\Pi a{:}A.B$, with $b{:}B$, B can be a term involving a!

fa' will only be a (valid) term if f and a' can be typed as $f{:}(\Pi a{:}A.B)$ and $a'{:}A$.

Naive semantics for 'Π': example: if $I = \{1, 2, 3\}$,
then $\Pi i{:}I.A_i = \prod_{i \in I} A_i =$
$= \{\{(1, a_1), (2, a_2), (3, a_3)\} \mid a_1 \in A_1, a_2 \in A_2, a_3 \in A_3\}$
$\cong A_1 \times A_2 \times A_3$

Introduction to CC (2) - example of a valid derivation

$$
\dfrac{
  \dfrac{\vdash *{:}\Box}{X{:}* \vdash X{:}*}\,\mathrm{s}_\Box
  \qquad
  \dfrac{
    \dfrac{\vdash *{:}\Box \quad \vdash *{:}\Box}{X{:}* \vdash *{:}\Box}\,\mathrm{weak}_\Box
    \quad
    \dfrac{\vdash *{:}\Box}{X{:}* \vdash X{:}*}\,\mathrm{s}_\Box
  }{X{:}*, x{:}X \vdash *{:}\Box}\,\mathrm{weak}_*
}{
  \dfrac{X{:}* \vdash (\Pi x{:}X.*){:}\Box}{X{:}*, F{:}(\Pi x{:}X.*) \vdash F{:}(\Pi x{:}X.*)}\,\mathrm{s}_\Box
}\,\Pi_{*\Box}
$$

A tree of (pre-)judgements
The leaves are axioms
The root is the conclusion
The bars are applications of the rules (coming soon!)
Most of the tree is “bureaucracy” (weakings, for example)

$$\underbrace{x_1 \overbrace{:A_1}^{\text{typing}}, \dots, x_n \overbrace{:A_n}^{\text{typing}}}_{\text{context}} \vdash t \overbrace{:T}^{\text{typing}}$$

The *terms* are the pre-terms that can appear as ‘x_i’s, ‘A_i’s, t or T.

The rules of the Calculus of Constructions

Rules of the system: ax, s, Π, λ, app, weak, conv
(conv is not shown; if $a{:}A$ and $A =_\beta A'$ then $a{:}A'$)

$$\begin{array}{ll}
\dfrac{\Gamma \vdash A{:}R \quad \Gamma, a{:}A \vdash B{:}S}{\Gamma \vdash (\Pi a{:}A.B){:}S}\ \Pi_{RS} & \dfrac{}{\vdash *{:}\Box}\ \mathrm{ax} \\
\dfrac{\Gamma, a{:}A \vdash b{:}B \quad \Gamma \vdash (\Pi a{:}A.B){:}S}{\Gamma \vdash (\lambda a{:}A.b){:}(\Pi a{:}A.B)}\ \lambda_{RS} & \dfrac{\Gamma \vdash A{:}R}{\Gamma, a{:}A \vdash a{:}A}\ \mathrm{s}_R \\
\dfrac{\Gamma \vdash f{:}(\Pi a{:}A.B) \quad \Gamma \vdash a'{:}A}{\Gamma \vdash fa'{:}B[a := a']}\ \mathrm{app}_{RS} & \dfrac{\Gamma, \Delta \vdash c{:}C \quad \Gamma \vdash A{:}R}{\Gamma, a{:}A, \Delta \vdash c{:}C}\ \mathrm{weak}_R
\end{array}$$

R, S sorts; a variable; Γ, Δ contexts
All other letters (a', A, b, B, c, C, f) represent terms
(Remember: even in a pre-judgement we can't have repeated declarations for the same variable)

Some facts (i.e., theorems) about CC
Each term of CC is either a sort, or a member of a sort, or a member of a member or a sort.

Each variable is a member of a member of a sort.

If $x_1{:}A_1, \ldots, x_n{:}A_n \vdash t{:}T$ then
the list $x_1, \ldots, x_n$ contains all the free variables of t and T;
there exist sorts $S_1, \ldots, S_n$ such that
$\vdash A_1{:}S_1$,
$x_1{:}A_1 \vdash A_2{:}S_2$,
$x_1{:}A_1, x_2{:}A_2 \vdash A_3{:}S_3$,
$\ldots$,
and if T is not a sort then there exists a sort S s.t.:
$x_1{:}A_1, \ldots, x_n{:}A_n \vdash T{:}S$.

Reading a pre-judgement aloud
Examples:
$\vdash (\lambda A{:}*.\lambda a{:}A.a){:}(\Pi A{:}*.\Pi a{:}A.A)$
$I{:}*, A{:}(\Pi i{:}I.*) \vdash (\Pi i{:}I.Ai){:}*$

More facts about CC: “classes” of terms, variations of the rules

If we know the “class” of each variable we know the class of every term:

$(\lambda a^{R\text{-}2}{:}A^{R\text{-}1}.b^{S\text{-}2})^{S\text{-}2}$ $\qquad (f^{S\text{-}2}a^{R\text{-}2})^{S\text{-}2}$

$(\Pi a^{R\text{-}2}{:}A^{R\text{-}1}.B^{S\text{-}1})^{S\text{-}1}$

		operator	:	kind	:	$\Box$
element	:	set	:	$*$		

a, a'	:	A	:	$*$			Π_{**},
b, fa'	:	B	:	$*$			λ_{**},
$f, (\lambda a{:}A.b)$	:	$(\Pi a{:}A.B)$	:	$*$			app_{**}
		a, a'	:	A	:	$\Box$	$\Pi_{\Box *}$,
b, fa'	:	B	:	$*$			$\lambda_{\Box *}$,
$f, (\lambda a{:}A.b)$	:	$(\Pi a{:}A.B)$	:	$*$			$\text{app}_{\Box *}$
a, a'	:	A	:	$*$			$\Pi_{* \Box}$,
		b, fa'	:	B	:	$\Box$	$\lambda_{* \Box}$,
		$f, (\lambda a{:}A.b)$	:	$(\Pi a{:}A.B)$	:	$\Box$	$\text{app}_{* \Box}$
		a, a'	:	A	:	$\Box$	$\Pi_{\Box\Box}$,
		b, fa'	:	B	:	$\Box$	$\lambda_{\Box\Box}$,
		$f, (\lambda a{:}A.b)$	:	$(\Pi a{:}A.B)$	:	$\Box$	$\text{app}_{\Box\Box}$

From CC with minimal contexts to ND and back

Definition: a “good” derivation tree is one in which:

- each variable appears always with the same type
- whenever there are two subtrees “deriving the same term”, they are equal

Take a good tree with minimal contexts, for example,

$$\dfrac{\dfrac{\dfrac{}{\vdash *{:}\Box}}{I{:}* \vdash I{:}*}\,s_\Box \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash *{:}\Box}}{I{:}* \vdash I{:}*}\,[\mathrm{s}_\Box]^{C2} \quad \dfrac{}{\vdash *{:}\Box}}{I{:}* \vdash (\Pi i{:}I.*){:}\Box}\,\Pi_{*\Box}}{I{:}*, A{:}(\Pi i{:}I.*) \vdash A{:}(\Pi i{:}I.*)}\,\mathrm{s}_\Box \quad \dfrac{\dfrac{\dfrac{}{\vdash *{:}\Box}}{I{:}* \vdash I{:}*}\,[\mathrm{s}_\Box]^{C1}}{I{:}*, i{:}I \vdash i{:}I}\,[\mathrm{s}_*]^2}{I{:}*, A{:}(\Pi i{:}I.*), i{:}I \vdash Ai{:}*}\,\mathrm{app}_{*\Box};1}{\dfrac{I{:}*, A{:}(\Pi i{:}I.*) \vdash (\Pi i{:}I.Ai){:}*}{I{:}*, A{:}(\Pi i{:}I.*), f{:}(\Pi i{:}I.Ai) \vdash f{:}(\Pi i{:}I.Ai)}\,s_*}\,\Pi_{**};2$$

erase its contexts, and put in the dictionary all the remaining typings:

$$\dfrac{\dfrac{\dfrac{}{*:\Box}}{I{:}*}\,s_\Box \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{*:\Box}}{I{:}*}\,[\mathrm{s}_\Box]^{C2} \quad \dfrac{}{*:\Box}}{(\Pi i{:}I.*){:}\Box}\,\Pi_{*\Box}}{A{:}(\Pi i{:}I.*)}\,\mathrm{s}_\Box \quad \dfrac{\dfrac{\dfrac{}{*:\Box}}{I{:}*}\,[\mathrm{s}_\Box]^{C1}}{i{:}I}\,[\mathrm{s}_*]^2}{Ai{:}*}\,\mathrm{app}_{*\Box};1}{\dfrac{(\Pi i{:}I.Ai){:}*}{f{:}(\Pi i{:}I.Ai)}\,s_*}\,\Pi_{**};2$$

Dictionary:
$i : I : *$
$A : (\Pi i{:}I.*) : \Box$
$Ai : *$
$f : (\Pi i{:}I.Ai) : *$

From CC with minimal contexts to ND and back (2)
...now erase the typings too:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{*}}{I}\;s_\square \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{*}}{I}\;[\mathrm{s}\square]^{C2} \quad \dfrac{}{*}}{\Pi i{:}I.*}\;\Pi_{*\square}}{A}\;\mathrm{s}\square \qquad \dfrac{\dfrac{\dfrac{}{*}}{I}\;[\mathrm{s}\square]^{C1}}{i}\;[\mathrm{s}_*]^2}{Ai}\;\mathrm{app}_{*\square};1}{\Pi i{:}I.Ai}\;\Pi_{**};2}{f}\;s_*$$

Dictionary:
$i : I : *$
$A : (\Pi i{:}I.*) : \square$
$Ai : *$
$f : (\Pi i{:}I.Ai) : *$

Fact: it is possible to reconstruct the original tree from this one.
New variables are introduced only at the "s" rules.
To help keep track of what's happening, we mark discharges with '$[\cdot]^n$'s and "contractions" with '$[\cdot]^{Cn}$'s.

It is also possible to discard the dictionary and reconstruct it just from that last tree.

Next step (not shown): replace the terms in a tree with names for them in the GP notation; let the dictionary translate some terms as applications — example, $b \dashrightarrow (a \mapsto b)(a)$.

Representing propositions and proofs in CC
Props is a sort; in CC, Props $= *$
(in other PTSs we may have Props $\neq *$ and we may be able to formalize *irrelevance of proofs*)

$\mathrm{prop}[P]$ is P seen as a proposition.
Naively, $\mathrm{prop}[P]$ is the set of proofs of P.
$\mathrm{prf}[P] : \mathrm{prop}[P] : \mathrm{Props} \quad (= *)$

Dictionary:

$$\begin{array}{rcl}
\mathrm{prop}[P \supset Q] & \dashrightarrow & (\Pi \mathrm{prf}[P]{:}\mathrm{prop}[P].\mathrm{prop}[Q]) \\
\mathrm{prf}[P \supset Q] & \dashrightarrow & (\mathrm{prf}[P] \mapsto \mathrm{prf}[Q]) \\
\mathrm{prop}[\forall x.P(x)] & \dashrightarrow & (\Pi x{:}\mathbf{E}[x].\mathrm{prop}[P(x)]) \\
\mathrm{prf}[\forall x.P(x)] & \dashrightarrow & (x \mapsto \mathrm{prf}[P(x)])
\end{array}$$

Terms for equality
To speak "internally" about equality between members of the same set we need a package of terms, that will be added to contexts...

equality$_*$ is a package consisting of:

eq$_*$	$(\mathrm{eq}_* A a a') : \mathrm{Props}$ $\mathrm{prop}[a = a'] \dashrightarrow \mathrm{eq}_* A a a'$
refl$_*$	$(\mathrm{refl}_* A a) : \mathrm{prop}[a = a]$ $\mathrm{prf}[a = a] \dashrightarrow \mathrm{refl}_* A a$
sim$_*$	$(\mathrm{sim}_* A a a') : \mathrm{prop}[a = a' \supset a' = a]$ $\mathrm{prf}[a = a' \supset a' = a] \dashrightarrow (\mathrm{sim}_* A a a')$
trans$_*$	$(\mathrm{trans}_* A a a' a'') : \mathrm{prop}[a = a' \supset (a' = a'' \supset a = a'')]$ $\mathrm{prf}[a = a' \supset (a' = a'' \supset a = a'')] \dashrightarrow (\mathrm{trans}_* A a a' a'')$

f$'$s.r.equality$_{**}$ is a package of terms saying that
all functions $f{:}(\Pi a{:}A.B)$ "respect equality".
This is a bit trickier, as it involves the "conv" rule.

Note: $a, a', a''{:}A{:}*$, $B{:}*$

Protocategories
The prefix "*proto*" will always mean "*without the terms for equalities*".

A protocategory $\mathbf{C} : \mathrm{Protocats}_{SR}$
(where S, R are sorts — typically $S = \Box$, $R = *$)
is a package containing

$$\begin{array}{ll}
\mathrm{Objs}_{\mathbf{C}} : S & \\
\mathrm{Hom}_{\mathbf{C}} & \mathrm{Hom}_{\mathbf{C}}\,AB : R \\
\mathrm{id}_{\mathbf{C}} & \mathrm{id}_{\mathbf{C}}A : \mathrm{Hom}_{\mathbf{C}}\,AA \\
\circ_{\mathbf{C}} & \circ_{\mathbf{C}}ABC(a \mapsto b)(b \mapsto c) : \mathrm{Hom}_{\mathbf{C}}\,AC
\end{array}$$

$(A, B, C : \mathrm{Objs}_{\mathbf{C}})$

Notational trick: $a \mapsto b : \mathbf{E}[a \mapsto b]$,
and the dictionary translates $\mathbf{E}[a \mapsto b] \dashrightarrow \mathrm{Hom}_{\mathbf{C}}\,AB$.

A protocategory comes without the 'prf' terms that say the $\circ_{\mathbf{C}}$ behaves well.
A category is a protocategory plus such terms.

A very special example: Set as a □∗-protocategory

$\mathbf{Set} : \mathrm{Protocats}_{\Box *}$
$\mathrm{Objs}_{\mathbf{Set}} = * : \Box$
$\mathrm{Hom}_{\mathbf{Set}}\, \mathbf{E}[x]\mathbf{E}[y] = \mathbf{E}[x \mapsto y] : *$
$\mathrm{Hom}_{\mathbf{Set}} := \lambda X{:}\,*\,.\lambda Y{:}\,*\,.(\Pi x{:}X.Y)$
$\mathrm{id}_{\mathbf{Set}}\mathbf{E}[x] = x \mapsto x$
$\mathrm{id}_{\mathbf{Set}} := \lambda X{:}\,*\,.\lambda x{:}X.x$
$\circ_{\mathbf{Set}}\mathbf{E}[x]\mathbf{E}[y]\mathbf{E}[z](x \mapsto y)(y \mapsto z) = (y \mapsto z) \circ (x \mapsto y)$
$\circ_{\mathbf{Set}} := \lambda X{:}\,*\,.\lambda Y{:}\,*\,.\lambda Z{:}\,*\,.$
$\qquad\qquad \lambda f{:}(\mathrm{Hom}_{\mathbf{Set}}\, XY).\lambda g{:}(\mathrm{Hom}_{\mathbf{Set}}\, YZ).$
$\qquad\qquad\quad \lambda x{:}X.g(fx)$

A typical $\Box *$-protocategory

$\mathrm{Objs}_{\mathbf{C}} : \Box$

$\mathrm{Hom}_{\mathbf{C}} : \Pi \mathbf{O}[a]{:}\mathrm{Objs}_{\mathbf{C}}.\Pi \mathbf{O}[b]{:}\mathrm{Objs}_{\mathbf{C}}.*$

$\mathbf{O}[a], \mathbf{O}[b], \mathbf{O}[c] : \mathrm{Objs}_{\mathbf{C}}$

like $\mathbf{E}[x], \mathbf{E}[y], \mathbf{E}[z] : \mathrm{Objs}_{\mathbf{Set}} = *$

but <u>we don't have a semantics</u> for "a", "b", "c"!!!

Just for $\mathbf{O}[a]$, $\mathbf{O}[b]$, $\mathbf{O}[c]$, $(a \mapsto b)$, $(b \mapsto c)$, $(a \mapsto c)$.

$$\begin{tikzcd}
x \arrow[r, mapsto] \arrow[dr, mapsto] & y \arrow[d, mapsto] \\
& z
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, mapsto] \arrow[dr, mapsto] & b \arrow[d, mapsto] \\
& c
\end{tikzcd}$$

in **Set** in **C**

New "operator" in the dictionary: **Cat** —

$\mathbf{Cat}[x] \dashrightarrow \mathbf{Set}$, $\mathbf{Cat}[a] \dashrightarrow \mathbf{C}$ $(\mathbf{O}[a]{:}\mathrm{Objs}_{\mathbf{C}})$

$x \mapsto y : \mathbf{E}[x \mapsto y] = \mathbf{E}[x] \to \mathbf{E}[y]$

$a \mapsto b : \mathbf{E}[a \mapsto b] = \mathrm{Hom}_{\mathbf{C}}\, \mathbf{O}[a] \mathbf{O}[b]$

System DNC: rules

$$\frac{\mathbf{O}[a] \quad a \Rightarrow a^F}{\mathbf{O}[a^F]}\;(\Rightarrow E_O) \qquad \frac{a' \mapsto a'' \quad a \Rightarrow a^F}{a'^F \mapsto a''^F}\;(\Rightarrow E_M)$$

$$\frac{\mathbf{O}[a] \quad a \overset{\bullet}{\to} (a^F \mapsto a^G)}{a^F \mapsto a^G}\;(\overset{\bullet}{\to} E)$$

Several "ren" rules, e.g.,

$$\frac{a \Rightarrow a^F}{\mathbf{O}[a \Rightarrow a^F]}\;\mathrm{ren} \qquad \frac{a \overset{\bullet}{\to} (a^F \mapsto a^G)}{(a \Rightarrow a^F) \mapsto (a \Rightarrow a^G)}\;\mathrm{ren}$$

Discharges are written in usual ND style:

$$\cfrac{\cfrac{\cfrac{\mathrm{Objs}_{\mathbf{C}}}{\mathbf{O}[a]}\;[\mathrm{s}]^1 \quad \ldots}{\mathbf{O}[a^F]}}{\mathbf{O}[a] \mapsto \mathbf{O}[a^F]}\;1 \quad \rightsquigarrow \quad \cfrac{\cfrac{[\mathbf{O}[a]]^1 \quad \ldots}{\mathbf{O}[a^F]}}{\mathbf{O}[a] \mapsto \mathbf{O}[a^F]}\;1$$

but then we need to know that $\mathbf{O}[a]$:$\mathrm{Objs}_{\mathbf{C}}$ (dictionary!) and the trees may split into several...

System DNC: ($\Rightarrow I$), example

If we know the action of $x \Rightarrow (a \mapsto x)$ on objects, i.e.,

$$\cfrac{\cfrac{\cfrac{\mathbf{E}[a] \qquad \cfrac{[\mathbf{O}[x]]^1}{\mathbf{E}[x]}\;\text{ren}}{\mathbf{E}[a \mapsto x]}}{\mathbf{O}[a \mapsto x]}\;\text{ren}}{\mathbf{O}[x] \mapsto \mathbf{O}[a \mapsto x]}\;1$$

then:

$$\cfrac{\cfrac{\cfrac{\cfrac{[a \mapsto x']^1 \qquad [x' \mapsto x'']^2}{a \mapsto x''}}{(a \mapsto x') \mapsto (a \mapsto x'')}\;1}{(x' \mapsto x'') \mapsto ((a \mapsto x') \mapsto (a \mapsto x''))}\;2}{x \Rightarrow (a \mapsto x)}\;\text{“}(\Rightarrow I)\text{”}$$

Note that we are constructing a proto-functor —
we are leaving out the “prf” terms!

System DNC: ($\overset{\bullet}{\to} I$), example

If we know $x \Rightarrow (b \mapsto x)$ and $x \Rightarrow (a \mapsto x)$, then the rule for creating
$x \overset{\bullet}{\to} ((b \mapsto x) \mapsto (a \mapsto x))$
is essentially the same as the rule for creating
$\mathbf{O}[x] \mapsto ((b \mapsto x) \mapsto (a \mapsto x))$
— again, we leave out the "prf" terms, and we only build a proto-natural transformation.

Example:

$$\dfrac{\dfrac{\dfrac{a \mapsto b \quad [b \mapsto x]^1}{a \mapsto x}}{(b \mapsto x) \mapsto (a \mapsto x)}\;1}{x \overset{\bullet}{\to} ((b \mapsto x) \mapsto (a \mapsto x))}\;2 \qquad \dfrac{\dfrac{\dfrac{a \mapsto b \quad \dfrac{\dfrac{\mathbf{O}[b] \quad [\mathbf{O}[x]]^2}{\mathbf{E}[b \mapsto x]}}{b \mapsto x}\;[s]^1}{a \mapsto x}}{(b \mapsto x) \mapsto (a \mapsto x)}\;1}{x \overset{\bullet}{\to} ((b \mapsto x) \mapsto (a \mapsto x))}\;2$$

Future work
Is DNC strongly normalizing?
Problems with this question:

- Right now DNC is just a layer of syntax on top of CC
- We need to formalize better how the dictionary works
- Some rules (e.g. $(\Rightarrow E)$) involve a certain amount of pattern-matching
- Reductions (cut-elimination) may require non-trivial renamings
- Which names are valid? DNC terms are inherently ambiguous...

DNC is not well-defined enough yet!!

What comes before studying SN for DNC

We need to learn more about DNC's ambiguities:
If a DNC term α has a natural construction,

- is it essentially unique?
- does it lift from the "syntactical world" to the "real world"?
- coherence/parametricity

Missing: rules for defining new categories

DNC and GP are tools for expressing the (intuitive) skeletons of some proofs (more precisely, the T-parts of proofs). So, morally, we should be able to represent nicely in GP/DNC:

- SN proofs for CC
- Categorical models for CC and other type theories
- "Basic category theory" (everything that a categorist would say is trivial)
- A lot about fibrations
- And also:
  - Topos logic (and a ND for it!)
  - Terms involving infinitesimals and derivatives (my original motivation, years ago)

Notes for people downloading these slides

In a very near future (before the end of June, I hope) I'm going to add notes about how I handwaved over these slides in the presentation, and, even more important, how to "pronounce" the DNC terms and diagrams... and also pointers to the literature, thanks to the people that discussed these ideas with me and helped me shape the presentation (Robert Seely, Jeff Egger, Robin Cockett), and a few other things.

Now I have to hurry up and prepare the slides for my presentation at the CMS 2002, in Quebec city... it's going to be only 15 mins long (this one was 45 mins) so I'll have to use a different approach, with more motivation and less details...

So: please check `http://angg.twu.net/` again in a few weeks for a new version of this. Sorry for the mess! Edrx, 2002jun13