Yoneda : Conversion for Dedukti or Automation for Coq ?

1. Short: This [1] solves some question of Ahrens [2] and Kan-Riehl [3], which is how to program Kelly's <<enriched categories>> and how the inter-dependence of <<naturality>> with <<category>> is cyclic. Also This [4] attempts to clarify the contrast <<categorical algebra>> (ring/locale-presentation and its "internal logic"), from <<categorial logic>> in the style of the <<enriched/encoded/programmed/recursion>> categories of Kelly-Dosen or Lawvere-Lambek and as attempted in [5], for example : the yoneda lemma and most categorial lemmas are no-more-than Gentzen's constructive logic of re-arranging the input-output positions <<modulo naturality>>. Now homotopy/knots/proof-nets may be held as (faithfull or almost-faithfull) semantical techniques (<<descent>>) to do this <<categorial logic>>, and the homotopy itself may be programmed in specialized grammars (for example [6] or HOTT).

2. The common assumption that catC(- , X) is dual to catC(Y , -) is FALSIFIED. This falsification originates from the description of the composition as some binary form instead of as some functional form which is programmed/encoded/enriched onto the computer. Then get some new thing which is named <<polymorphism>> from which to define <<polymorph category>>. This is the only-ever real description and deduction of the yoneda lemma, which says that the image of polyF (which is injective and contained in natural transformations) also contains all natural transformations.

3. Some polymorph category is given by polyF, which is commonly (_1 o> _2), polymorph in V and polymorph in A :

```
Variable obF : Type.
Variable polyF00 : obF -> obF -> obV.
Notation "F[0 A1 ~> A2 ]0" := (polyF00 A1 A2) (at level 25).
Parameter polyF : forall (B : obF), forall (V : obV) (A : obF),
        V(0 V |- F[0 B ~> A ]0 )0 ->
        forall X : obF, V(0 F[0 A ~> X ]0  |- [0 V ~> F[0 B ~> X ]0 ]0 )0.
```

4. And to get polymorph functor, instead of describing F : catA --> catB then (contrast yoneda structures) describe catV[V , catB[B , F -]] : catA --> catV , more precisely

```
  Variable polyF0 : obA -> obB.
  Notation "F|0 A" := (polyF0 A) (at level 4, right associativity).
  Notation "F[0 B ~> A ]0" := (B[0 B ~> F|0 A ]0) (at level 25).
  Parameter polyF : forall (V : obV) (B : obB) (A : obA),
        V(0 V |- F[0 B ~> A ]0 )0 ->
        forall X : obA, V(0 A[0 A ~> X ]0  |- [0 V ~> F[0 B ~> X ]0 ]0 )0.
```

5. And to get polymorph transformation, instead of describing phi A : G A -> H A then a-la-dosen (contrast weighted colimiting Kan extension) describe phi _f : catV(V , catB[B , G A]) -> catV(V , catB[B , H A]) , more precisely

```
   Parameter poly_phi : forall (V : obV) (B : obB) (A : obA),
                      V(0 V |- F[0 B ~> A ]0 )0 ->
                      V(0 V |- G[0 B ~> A ]0 )0 .
```

And finally one shall relate the earlier <<naturality of transformation inside catV>> to this new <<polymorphism>> of transformation.

6. The earlier texts referring to Maclane associativity coherence and Dosen semiassociativity coherence and Dosen cut elimination for adjunctions and Chlipala ur/web database programming are all related to this present text which is how to program Borceux logically-enriched categories.

[1] 1337777.OOO, https://github.com/1337777/borceux/blob/master/borceuxSolution.v

[2] Ahrens, https://github.com/benediktahrens/monads/blob/trunk/CAT/enriched_cat.v

[3] Riehl, http://www.math.jhu.edu/~eriehl/context.pdf

[4] 1337777.OOO, https://github.com/1337777/borceux/blob/master/chic05.pdf

[5] 1337777.OOO, https://github.com/1337777/dosen/blob/master/itp.pdf

[6] Ye, http://katherineye.com/post/129960474471/strange-loops-capturing-knots-with-powerful