```
(*****************************************************************************************************************************)
(* Reification:                                                                                                              *)
(*                                                                                                                           *)
(*   A reification is a functor R from one enrichING category A to another enrichING category B which, for every X, forms      *)
(*   a commuting square with Hom(X,-):a->A and Hom(I,-):b->B (up to natural isomorphism).                                     *)
(*                                                                                                                           *)
(*****************************************************************************************************************************)

Generalizable All Variables.
Require Import Preamble.
Require Import General.
Require Import Categories_ch1_3.
Require Import Functors_ch1_4.
Require Import Isomorphisms_ch1_5.
Require Import ProductCategories_ch1_6_1.
Require Import OppositeCategories_ch1_6_2.
Require Import Enrichment_ch2_8.
Require Import Subcategories_ch7_1.
Require Import NaturalTransformations_ch7_4.
Require Import NaturalIsomorphisms_ch7_5.
Require Import BinoidalCategories.
Require Import PreMonoidalCategories.
Require Import MonoidalCategories_ch7_8.
Require Import Coherence_ch7_8.
Require Import Enrichments.
Require Import Enrichment_ch2_8.
Require Import RepresentableStructure_ch7_2.

Opaque HomFunctor.
Opaque functor_comp.
Structure Reification (K:Enrichment) (C:Enrichment) (CI:C) :=
{ reification_r_obj     : K -> K -> C
; reification_rstar_obj : enr_v K -> enr_v C
; reification_r         : forall k:K, Functor K C (reification_r_obj k)
; reification_rstar_f   :             Functor (enr_v K) (enr_v C) reification_rstar_obj
; reification_rstar     :             PreMonoidalFunctor (enr_v_mon K) (enr_v_mon C) reification_rstar_f
; reification_commutes  : ∀ k, reification_r k >>>> HomFunctor C CI <~~~> HomFunctor K k >>>> reification_rstar_f

(* We require that the host language (but NOT the guest language) be pure, i.e. all morphisms central, to simplify
 * things.  If this doesn't suit you, just consider the "host language" here to be the pure sublanguage of the
 * host language, and toss on the inclusion functor to the full language *)
```

```
; reification_host_lang_pure   : CommutativeCat (enr_c_pm C)
}.
Transparent HomFunctor.
Transparent functor_comp.

Coercion reification_rstar : Reification >-> PreMonoidalFunctor.
Implicit Arguments Reification                [ ].
Implicit Arguments reification_r_obj          [ K C CI ].
Implicit Arguments reification_r              [ K C CI ].
Implicit Arguments reification_rstar          [ K C CI ].
Implicit Arguments reification_rstar_f        [ K C CI ].
Implicit Arguments reification_commutes       [ K C CI ].
Implicit Arguments reification_rstar_obj      [ K C CI ].
```