```
(*****************************************************************************************************************************)
(* ReificationsAndGeneralizedArrows:                                                                                         *)
(*                                                                                                                           *)
(*   For each pair of enrichments E1 and E2, there is a bijection between the generalized arrows E1->E2 and the reifications     *)
(*   E1->E2                                                                                                                  *)
(*                                                                                                                           *)
(*****************************************************************************************************************************)

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 MonoidalCategories_ch7_8.
Require Import PreMonoidalCategories.
Require Import Coherence_ch7_8.
Require Import Enrichment_ch2_8.
Require Import Enrichments.
Require Import RepresentableStructure_ch7_2.
Require Import Reification.
Require Import GeneralizedArrow.
Require Import GeneralizedArrowFromReification.
Require Import ReificationFromGeneralizedArrow.
Require Import ReificationCategory.
Require Import GeneralizedArrowCategory.

Section ReificationsEquivalentToGeneralizedArrows.

  Ltac if_transitive :=
    match goal with [ |- ?A ≃ ?B ] => refine (@if_comp _ _ _ _ _ _ _ A _ _ _ B _ _)
    end.

  Lemma roundtrip_lemma
    `{D:Category}`{E:Category}
```

```
  {Gobj}{G:Functor E D Gobj} G_full G_faithful `{C:Category}{Fobj}(F:Functor C (FullImage G) Fobj) :
  (F >>>> (ff_functor_section_functor G G_full G_faithful >>>> G)) ≃ (F >>>> FullImage_InclusionFunctor _).
  if_transitive.
    eapply if_inv.
    apply (if_associativity F (ff_functor_section_functor G _ _) G).
  if_transitive.
    apply (if_associativity F (ff_functor_section_functor G _ _) G).
    apply if_respects.
      apply if_id.
      if_transitive; [ idtac | apply if_left_identity ].
  apply (if_comp(F2:=(ff_functor_section_functor G G_full G_faithful) >>>> RestrictToImage G >>>> FullImage_InclusionFunctor _)).
  apply if_inv.
  apply (if_associativity (ff_functor_section_functor G G_full G_faithful)
    (RestrictToImage G) (FullImage_InclusionFunctor G)).
  apply if_respects.
  apply ff_functor_section_splits_niso.
  apply if_id.
  Qed.

Definition step1_niso
  `(K:SurjectiveEnrichment) `(C:MonicEnrichment ce) (CM:MonoidalEnrichment ce) (reification : Reification K ce (enr_c_i ce))
  : reification_rstar reification ≃ (RestrictToImage reification >>>> R' K CM reification >>>> FullImage_InclusionFunctor CM).
  exists (fun c1 => homset_tensor_iso K CM reification c1).
  intros.
  simpl.
  etransitivity.
  eapply comp_respects; [ apply reflexivity | idtac ].
  apply associativity.
  apply iso_comp1_right.
  Qed.

Lemma roundtrip_garrow_to_garrow
  `(K:SurjectiveEnrichment) `(C:MonicEnrichment ce) (CM:MonoidalEnrichment ce) (garrow : GeneralizedArrow K ce)
  : garrow ≃ garrow_from_reification K C CM (reification_from_garrow K CM garrow).
  apply if_inv.
  eapply if_comp.
  eapply if_inv.
  unfold garrow_from_reification.
  simpl.
  unfold mf_F.
  unfold garrow_functor.
```

```
  apply (if_associativity
    (RestrictToImage (reification_from_garrow K CM garrow))
    (R' K CM (reification_from_garrow K CM garrow))
    (step2_functor C)).

  apply (ffc_functor_weakly_monic _ me_full me_faithful me_conservative me_conservative).
  eapply if_comp.
  apply (if_associativity
    (RestrictToImage (reification_from_garrow K CM garrow) >>>> R' K CM (reification_from_garrow K CM garrow))
    (step2_functor C)
    me_homfunctor).

  set ((R' K CM (reification_from_garrow K CM garrow))) as f.
  unfold HomFunctor_fullimage in f.
  set (roundtrip_lemma (me_full(MonicEnrichment:=C))) as q.
  set (q (me_faithful(MonicEnrichment:=C)) _ _ _ _ f) as q'.

  eapply if_comp.
    apply (if_associativity (RestrictToImage (reification_from_garrow K CM garrow)) f (step2_functor C >>>> me_homfunctor)).
    unfold step2_functor.
    eapply if_comp.
    apply (if_respects
      (RestrictToImage (reification_from_garrow K CM garrow))
      (RestrictToImage (reification_from_garrow K CM garrow))
      _ _ (if_id _) q').

  eapply if_comp.
    eapply if_inv.
    apply (if_associativity (RestrictToImage (reification_from_garrow K CM garrow)) f (FullImage_InclusionFunctor me_homfunctor)).

  apply if_inv.
    apply (step1_niso K C CM (reification_from_garrow K CM garrow)).
    Qed.

Lemma roundtrip_reification_to_reification
  `(K:SurjectiveEnrichment) `(C:MonicEnrichment ce) (CM:MonoidalEnrichment ce) (reification : Reification K ce (enr_c_i ce))
  : reification ≃ reification_from_garrow K CM (garrow_from_reification K C CM reification).

  simpl.
    unfold garrow_functor.
```

```
    eapply if_comp.
      apply (step1_niso K C CM reification).

    unfold garrow_monoidal.
      unfold mf_F.
      apply if_inv.
      eapply if_comp.
      apply (if_associativity (RestrictToImage reification) (R' K CM reification >>>> step2_functor C)
        (HomFunctor ce (pmon_I (enr_c_pm ce)))).
      apply if_inv.
      eapply if_comp.
      apply (if_associativity (RestrictToImage reification) (R' K CM reification)
        (FullImage_InclusionFunctor (HomFunctor ce (pmon_I (enr_c_pm ce))))).

    apply (if_respects (RestrictToImage reification) (RestrictToImage reification) _ _ (if_id _)).
      apply if_inv.
      eapply if_comp.
      apply (if_associativity  (R' K CM reification) (step2_functor C) (HomFunctor ce _)).

    set ((R' K CM reification)) as f.
      unfold HomFunctor_fullimage in f.
      set (roundtrip_lemma (me_full(MonicEnrichment:=C))) as q.
      set (q (me_faithful(MonicEnrichment:=C)) _ _ _ _ f) as q'.
      apply q'.
      Qed.

End ReificationsEquivalentToGeneralizedArrows.
```