```
(*****************************************************************************************************************************)
(* ReificationsIsomorphicToGeneralizedArrows:                                                                                *)
(*                                                                                                                           *)
(*   The category of generalized arrows and the category of reifications are isomorphic categories.                         *)
(*                                                                                                                           *)
(*****************************************************************************************************************************)

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 PreMonoidalCategories.
Require Import MonoidalCategories_ch7_8.
Require Import Coherence_ch7_8.
Require Import RepresentableStructure_ch7_2.
Require Import Reification.
Require Import GeneralizedArrow.
Require Import GeneralizedArrowFromReification.
Require Import ReificationFromGeneralizedArrow.
Require Import ReificationCategory.
Require Import GeneralizedArrowCategory.
Require Import ReificationCategory.
Require Import ReificationsAndGeneralizedArrows.
Require Import WeakFunctorCategory.
Require Import BijectionLemma.
Require Import Enrichments.

Section ReificationsIsomorphicToGeneralizedArrows.

    Definition M1 {c1 c2 : SMMEs} :
      (c1 ~~{ MorphismsOfCategoryOfGeneralizedArrows }~~> c2) ->
      (c1 ~~{ MorphismsOfCategoryOfReifications }~~> c2).
      intro GA.
```

```
  destruct GA; [ apply roi_id | idtac ].
  apply roi_reif.
  apply (reification_from_garrow s1 s2 g).
  Defined.

(* I tried really hard to avoid this *)
Require Import Coq.Logic.Eqdep.

Inductive Heq : forall {A}{B}, A -> B -> Prop :=
  heq : forall {A} (a:A), Heq a a.

Lemma invert_ga' : forall (a b: SMME)
  (f:a~~{MorphismsOfCategoryOfGeneralizedArrows}~~>b), a=b ->
  (Heq f (gaoi_id a)) \/ (exists f', Heq f (gaoi_ga a b f')).
  intros.
  destruct f.
  left; apply heq.
  subst; right.
  exists g.
  apply heq.
  Defined.

Lemma invert_ga : forall (a: SMME)
  (f:a~~{MorphismsOfCategoryOfGeneralizedArrows}~~>a),
  (f = gaoi_id _) \/ (exists f', f = gaoi_ga _ _ f').
  intros.
  set (invert_ga' a a f (refl_equal a)) as q.
  destruct q.
  left.
  inversion H.
  apply inj_pairT2 in H2.
  apply inj_pairT2 in H1.
  subst; auto.
  right.
  destruct H.
  exists x.
  inversion H.
  apply inj_pairT2 in H2.
  apply inj_pairT2 in H1.
  subst; auto.
  Qed.
```

```
Lemma invert_reif' : forall (a b: SMME)
  (f:a~~{MorphismsOfCategoryOfReifications}~~>b), a=b ->
  (Heq f (roi_id a)) \/ (exists f', Heq f (roi_reif a b f')).
  intros.
  destruct f.
  left; apply heq.
  subst; right.
  exists r.
  apply heq.
  Defined.

Lemma invert_reif : forall (a: SMME)
  (f:a~~{MorphismsOfCategoryOfReifications}~~>a),
  (f = roi_id _) \/ (exists f', f = roi_reif _ _ f').
  intros.
  set (invert_reif' a a f (refl_equal a)) as q.
  destruct q.
  left.
  inversion H.
  apply inj_pairT2 in H2.
  apply inj_pairT2 in H1.
  subst; auto.
  right.
  destruct H.
  exists x.
  inversion H.
  apply inj_pairT2 in H2.
  apply inj_pairT2 in H1.
  subst; auto.
  Qed.

Definition M1_Functor : Functor MorphismsOfCategoryOfGeneralizedArrows MorphismsOfCategoryOfReifications (fun x => x).
  refine {| fmor := fun a b f => M1 f |}.
  intros.
    unfold hom in *.
    unfold eqv in *.
    simpl in *.
    destruct f.
    set (invert_ga _ f') as q.
    destruct q; subst.
```

```
    apply if_id.
    simpl in *.
    destruct H0; subst.
    apply H.
    simpl in *.
    destruct f'; simpl in *.
    apply H.
    apply H.
  intros; simpl.
    apply if_id.
  intros.
    simpl.
    destruct f; simpl.
    apply if_id.
    destruct g; simpl.
    apply if_id.
    simpl.
    apply (if_associativity
      ((ga_functor g0 >>>> HomFunctor s0 (enr_c_i s0))) (ga_functor g) (HomFunctor s2 (enr_c_i s2))).
    Defined.

Definition M2 (c1 c2 : SMMEs) :
  (c1 ~~{ MorphismsOfCategoryOfReifications }~~> c2) ->
  (c1 ~~{ MorphismsOfCategoryOfGeneralizedArrows }~~> c2).
  intro RE.
  destruct RE; [ apply gaoi_id | idtac ].
  apply gaoi_ga.
  apply (garrow_from_reification s1 (smme_mee s2) s2 r).
  Defined.

Lemma eqv1 a b (f : a ~~{ MorphismsOfCategoryOfGeneralizedArrows }~~> b)
  (f' : a ~~{ MorphismsOfCategoryOfGeneralizedArrows }~~> b)
  (H : generalizedArrowOrIdentityFunc a b f ≃ generalizedArrowOrIdentityFunc a b f') :
  generalizedArrowOrIdentityFunc a b (M2 a b (M1 f)) ≃ generalizedArrowOrIdentityFunc a b f'.
    unfold hom in *.
    set (@roundtrip_garrow_to_garrow a b (smme_mee b) (smme_mon b)) as q.
    destruct f; simpl in *.
    apply H.
    apply if_inv.
    apply (if_comp (if_inv H)).
    clear H.
```

```
    apply (if_respects
      (ga_functor g)
      (garrow_functor s1 (smme_mee s2) s2 (reification_from_garrow s1 s2 g))
      (HomFunctor (senr_c s2) (senr_c_i s2))
      (HomFunctor (senr_c s2) (senr_c_i s2))
    ).
    apply q.
    apply if_id.
    Qed.

Lemma eqv2 a b (f : a ~~{ MorphismsOfCategoryOfReifications }~~> b)
  (f' : a ~~{ MorphismsOfCategoryOfReifications }~~> b)
  (H : reificationOrIdentityFunc a b f ≃ reificationOrIdentityFunc a b f') :
  reificationOrIdentityFunc _ _ (M1 (M2 _ _ f)) ≃ reificationOrIdentityFunc _ _ f'.
    unfold hom in *.
    set (@roundtrip_reification_to_reification a b (smme_mee b) (smme_mon b)) as q.
    destruct f; simpl.
    apply H.
    apply if_inv.
    apply (if_comp (if_inv H)).
    clear H.
    simpl.
    simpl in q.
    simpl in q.
    apply q.
    Qed.

Lemma M2_respects :
  forall a b (f f':a~~{MorphismsOfCategoryOfReifications}~~>b),
     f ~~ f' ->
     M2 a b f ~~ M2 a b f'.
  intros.
    unfold hom in *.
    unfold eqv in *.
    simpl in *.
    destruct f.
      set (invert_reif _ f') as q.
      destruct q; subst.
      apply if_id.

      simpl in *.
```

```
    destruct H0; subst.
    simpl in *.
    unfold garrow_functor.
    unfold step2_functor.
    apply (if_comp H).
    clear H.
    eapply if_comp.
    apply (step1_niso smme (smme_mee smme) (smme_mon smme) x).
    apply if_inv.
    apply if_inv.
    eapply if_comp.
    apply (if_associativity (RestrictToImage x) (R' smme smme x) (FullImage_InclusionFunctor _)).
    apply if_inv.
    eapply if_comp.
    apply (if_associativity (RestrictToImage x) ((R' smme smme x >>>>
             ff_functor_section_functor me_homfunctor me_full me_faithful))
    (HomFunctor (senr_c smme) (senr_c_i smme))).
    apply (if_respects (RestrictToImage x) (RestrictToImage x)).
    apply (if_id (RestrictToImage x)).
    unfold mf_F.
    eapply if_comp.
      apply (if_associativity (R' smme smme x) (ff_functor_section_functor me_homfunctor me_full me_faithful)
        (HomFunctor (senr_c smme) (senr_c_i smme))).
    set (roundtrip_lemma (me_full(MonicEnrichment:=smme_mee smme))) as q.
    set (R' smme smme x) as f.
    set (me_faithful(MonicEnrichment:=smme_mee smme)) as ff.
    unfold HomFunctor_fullimage in f.
    unfold mf_F in f.
    set (q ff _ _ (FullImage x) _ f) as q'.
    unfold me_homfunctor in q'.
    exact q'.

simpl in *.
  destruct f'; simpl in *.
  simpl in *.
  apply if_inv in H.
  eapply if_comp; [ idtac | eapply if_inv; apply H ].
  clear H.
  unfold garrow_functor.
  unfold step2_functor.
  apply if_inv.
```

```
  eapply if_comp.
  apply (step1_niso smme (smme_mee smme) (smme_mon smme) r).

  rename r into x.
  apply if_inv.
  eapply if_comp.
  apply (if_associativity (RestrictToImage x) ((R' smme smme x >>>>
           ff_functor_section_functor me_homfunctor me_full me_faithful))
  (HomFunctor (senr_c smme) (senr_c_i smme))).
  apply if_inv.
  eapply if_comp.
  apply (if_associativity (RestrictToImage x) (R' smme smme x) (FullImage_InclusionFunctor _)).
  apply (if_respects (RestrictToImage x) (RestrictToImage x)).
  apply (if_id (RestrictToImage x)).

  unfold mf_F.
  apply if_inv.

  eapply if_comp.
    apply (if_associativity (R' smme smme x) (ff_functor_section_functor me_homfunctor me_full me_faithful)
      (HomFunctor (senr_c smme) (senr_c_i smme))).
  set (roundtrip_lemma (me_full(MonicEnrichment:=smme_mee smme))) as q.
    set (R' smme smme x) as f.
    set (me_faithful(MonicEnrichment:=smme_mee smme)) as ff.
    unfold HomFunctor_fullimage in f.
    unfold mf_F in f.
    set (q ff _ _ (FullImage x) _ f) as q'.
    unfold me_homfunctor in q'.
    exact q'.

simpl in *.
  unfold garrow_functor.
  unfold step2_functor.
  set (step1_niso s1 (smme_mee s2) s2 r) as q.
  apply if_inv in q.
  eapply if_comp.
  eapply if_comp; [ idtac | apply q ].

  eapply if_comp.
  apply (if_associativity
    (RestrictToImage r)
```

```
  (R' s1 s2 r >>>> ff_functor_section_functor me_homfunctor me_full me_faithful)
  (HomFunctor (senr_c s2) (senr_c_i s2))).
apply if_inv.
eapply if_comp.
apply (if_associativity
  (RestrictToImage r)
  (R' s1 s2 r)
  (FullImage_InclusionFunctor _)).
apply (if_respects
  (RestrictToImage r)
  (RestrictToImage r)
  (R' s1 s2 r >>>> FullImage_InclusionFunctor _)
  (((R' s1 s2 r >>>> ff_functor_section_functor me_homfunctor me_full me_faithful) >>>>
     HomFunctor (senr_c s2) (senr_c_i s2)))).
apply (if_id _).
apply if_inv.
eapply if_comp.
apply (if_associativity
  (R' s1 s2 r)
  (ff_functor_section_functor me_homfunctor me_full me_faithful)
  (HomFunctor (senr_c s2) (senr_c_i s2))).
apply roundtrip_lemma.

apply if_inv.
set (step1_niso s1 (smme_mee s2) s2 r0) as q'.
apply if_inv in q'.
eapply if_comp.
eapply if_comp; [ idtac | apply q' ].
eapply if_comp.
apply (if_associativity
  (RestrictToImage r0)
  (R' s1 s2 r0 >>>> ff_functor_section_functor me_homfunctor me_full me_faithful)
  (HomFunctor (senr_c s2) (senr_c_i s2))).
apply if_inv.
eapply if_comp.
apply (if_associativity
  (RestrictToImage r0)
  (R' s1 s2 r0)
  (FullImage_InclusionFunctor _)).
apply (if_respects
  (RestrictToImage r0)
```

```
      (RestrictToImage r0)
      (R' s1 s2 r0 >>>> FullImage_InclusionFunctor _)
      (((R' s1 s2 r0 >>>> ff_functor_section_functor me_homfunctor me_full me_faithful) >>>>
        HomFunctor (senr_c s2) (senr_c_i s2)))).
    apply (if_id _).
    apply if_inv.
    eapply if_comp.
    apply (if_associativity
      (R' s1 s2 r0)
      (ff_functor_section_functor me_homfunctor me_full me_faithful)
      (HomFunctor (senr_c s2) (senr_c_i s2))).
    apply roundtrip_lemma.
    apply if_inv.
    apply H.
    Qed.

  Definition M2_Functor : Functor MorphismsOfCategoryOfReifications MorphismsOfCategoryOfGeneralizedArrows (fun x => x).
    refine {| fmor := fun a b f => M2 _ _ f |}.
    apply M2_respects.
    intros; simpl; apply if_id.
    intros; apply (@bijection_lemma _ _ _ _ _ M1_Functor M2); intros.
    apply M2_respects; auto.
    unfold fmor; simpl.
    apply (@eqv1 _ _ f0 f0).
    apply if_id.
    unfold fmor; simpl.
    apply (@eqv2 _ _ f0 f0).
    apply if_id.
    Defined.

  Theorem ReificationsAreGArrows : IsomorphicCategories CategoryOfGeneralizedArrows CategoryOfReifications.
    refine {| ic_f := M1_Functor ; ic_g := M2_Functor |}.
    unfold EqualFunctors; intros; apply heq_morphisms_intro; unfold eqv in *; simpl in *.
    apply (eqv1 _ _ f f'); auto.
    unfold EqualFunctors; intros; apply heq_morphisms_intro; unfold eqv in *; simpl in *.
    apply (eqv2 _ _ f f'); auto.
    Qed.

End ReificationsIsomorphicToGeneralizedArrows.
```