```
(*****************************************************************************************************************************)
(* CategoryOfGeneralizedArrows:                                                                                              *)
(*                                                                                                                           *)
(*   There is a category whose objects are surjective monic monoidal enrichments (SMME's) and whose morphisms                *)
(*   are generalized  Arrows                                                                                                 *)
(*                                                                                                                           *)
(*****************************************************************************************************************************)

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 Enrichment_ch2_8.
Require Import Enrichments.
Require Import RepresentableStructure_ch7_2.
Require Import GeneralizedArrow.
Require Import WeakFunctorCategory.

(*
 * Technically reifications form merely a *semicategory* (no identity
 * maps), but one can always freely adjoin identity maps (and nothing
 * else) to a semicategory to get a category whose non-identity-map
 * portion is identical to the original semicategory
 *
 * Also, technically this category has ALL enrichments (not just the
 * surjective monic monoidal ones), though there maps OUT OF only the
 * surjective enrichments and INTO only the monic monoidal
 * enrichments.  It's a big pain to do this in Coq, but sort of might
 * matter in real life: a language with really severe substructural
```

```
 * restrictions might fail to be monoidally enriched, meaning we can't
 * use it as a host language.  But that's for the next paper...
 *)
Inductive GeneralizedArrowOrIdentity : SMMEs -> SMMEs -> Type :=
  | gaoi_id   : forall smme:SMMEs,                              GeneralizedArrowOrIdentity smme smme
  | gaoi_ga : forall s1 s2:SMMEs, GeneralizedArrow s1 s2  -> GeneralizedArrowOrIdentity s1   s2.

Definition generalizedArrowOrIdentityFobj (s1 s2:SMMEs) (f:GeneralizedArrowOrIdentity s1 s2) : s1 -> s2 :=
  match f in GeneralizedArrowOrIdentity S1 S2 return S1 -> S2 with
  | gaoi_id  s       => fun x => x
  | gaoi_ga s1 s2 f  => fun a => ehom(ECategory:=s2) (enr_c_i (smme_e s2)) (ga_functor_obj f a)
  end.

Definition generalizedArrowOrIdentityFunc s1 s2 (f:GeneralizedArrowOrIdentity s1 s2)
  : Functor s1 s2 (generalizedArrowOrIdentityFobj _ _ f) :=
  match f with
  | gaoi_id  s       => functor_id _
  | gaoi_ga s1 s2 f  => ga_functor f >>>> HomFunctor s2 (enr_c_i s2)
  end.

Instance compose_generalizedArrows (s0 s1 s2:SMMEs)
  (g01:GeneralizedArrow s0 s1)(g12:GeneralizedArrow s1 s2) : GeneralizedArrow s0 s2 :=
    { ga_functor_monoidal := g01 >>⊗>> smme_mon s1 >>⊗>> g12 }.
  apply ga_host_lang_pure.
  Defined.

Definition generalizedArrowOrIdentityComp
  : forall s1 s2 s3, GeneralizedArrowOrIdentity s1 s2 -> GeneralizedArrowOrIdentity s2 s3 -> GeneralizedArrowOrIdentity s1 s3.
  intros.
  destruct X.
    apply X0.
  destruct X0.
    apply (gaoi_ga _ _ g).
    apply (gaoi_ga _ _ (compose_generalizedArrows _ _ _ g g0)).
    Defined.

Definition MorphismsOfCategoryOfGeneralizedArrows : @SmallFunctors SMMEs.
  refine {| small_func      := GeneralizedArrowOrIdentity
          ; small_func_id   := fun s => gaoi_id s
          ; small_func_func := fun smme1 smme2 f => generalizedArrowOrIdentityFunc _ _ f
          ; small_func_comp := generalizedArrowOrIdentityComp
```

```
        |}; intros; simpl.
  apply if_id.
  destruct f as [|fobj f]; simpl in *.
    apply if_inv.
    apply if_left_identity.
  destruct g as [|gobj g]; simpl in *.
    apply if_inv.
    apply if_right_identity.
  unfold mf_F.
  idtac.
  apply if_associativity.
  Defined.

Definition CategoryOfGeneralizedArrows :=
  WeakFunctorCategory MorphismsOfCategoryOfGeneralizedArrows.
```