```agda
{-
           IDENTITIES VIA IDEMPOTENT EQUIVALENCES
                  complementing the paper
   Internal ∞-Categorical Models of Dependent Type Theory

Summary. The standard definition of a semicategory consists
of objects, morphisms, composition, and associativity. How
can we say that a given semicategory has identities, i.e. is
a category? The standard answer is to say that the
semicategory has identities which satisfy left-neutrality and
right-neutrality.

In this short note, I present another definition of the
statement "has-identities" (a.k.a. "is-a-category").
The motivation is that I want "is-a-category" to be a
*property* of a semicategory rather than *data*, i.e. I want
it to be a proposition a.k.a. proof-irrelevant. For a semi-
category where the morphisms form a set, this is already the
case for the standard answer above. However, without this
additional assumption, the standard answer is data.

My new definition of "is-a-category" is "for every object x,
there is an endomorphism i ∈ Hom(x,x) such that i is
an idempotent equivalence". This Agda file shows that this
is a propositional property. I also show below that a semi-
category has idempotent equivalences if and only if it has
standard identities.

The Agda code can be type-checked with Agda 2.6.1 and makes
use of the library HoTT-Agda, using Andrew Swan's fork that
is 2.6.1-compatible:
https://github.com/awswan/HoTT-Agda/tree/agda-2.6.1-compatible
-}


{-# OPTIONS --without-K #-}

module Identities where
open import HoTT public
open import Iff

{-  A *wild semicategory*, sometimes also known as a *wild
    semigroupoid*, consists of:
      - Ob: a type of objects;
      - Hom: a type family of morphisms (for increased
        generality possibly in another universe than Ob, but
        it doesn't matter);
      - and an associative binary operation _⋄_.
    This is of course just a "category without identities".
    Note that we do *NOT* ask for set-truncation; Ob and Hom
    are arbitrary types/type families!
-}

record SemiCategory j₁ j₂ : Type (lsucc (lmax j₁ j₂)) where
  infixr 40 _⋄_
  field
    Ob  : Type j₁
    Hom : Ob → Ob → Type j₂
    _⋄_ : ∀ {x y z} → Hom y z → Hom x y → Hom x z
    ass : ∀ {x y z w} {f : Hom z w} {g : Hom y z} {h : Hom x y}
        → (f ⋄ g) ⋄ h == f ⋄ (g ⋄ h)

{-  For the rest of this file, we assume that C is a given
    fixed wild semicategory. We work inside a module which
    fixes C.
-}

module _ {j₁ j₂} (C : SemiCategory j₁ j₂) where
  open SemiCategory C
  j = lmax j₁ j₂

  {- A "standard identity" is a morphism which is left and
     right neutral. This is the normal, well-known definition.
  -}

  module _ {y : Ob} (i : Hom y y) where
    is-left-neutral = {x : Ob} (f : Hom x y) → i ⋄ f == f
    is-right-neutral = {z : Ob} (g : Hom y z) → g ⋄ i == g
    is-standard-id = is-left-neutral × is-right-neutral

  {- We say that a semicategory (here, the semicategory C)
     is a *standard category* if every object comes with
     a morphism which is left- and right-neutral. This is
     the usual definition of what it means to "have
     identities". -}

  is-standard-category = (x : Ob) → Σ (Hom x x) is-standard-id
```

```
{- The problem with these identities is that "having
   standard identities" is not a propositional property:
   it is structure, i.e. C can have standard identities in
   multiple different ways. This makes "having standard
   identities" ill-behaved, as demonstrated in the paper
   (see Example 9, initial model in the wild/incoherent
   setting).

   We now develop an alternative, and better, definition
   of identities, namely *idempotent equivalences*, and
   this will lead to a propositional property.
-}

is-idpt : {x : Ob} (f : Hom x x) → Type j₂
is-idpt f = f ⋄ f == f

{- Note: `is-idpt f` a.k.a. "f is idempotent" is not
   necessarily a proposition (as 'Hom x x' might not be a
   set). -}

is-eqv : {x y : Ob} (g : Hom x y) → Type j
is-eqv {x} {y} g =   ((z : Ob) → is-equiv λ (h : Hom y z) → h ⋄ g)
                   × ((w : Ob) → is-equiv λ (f : Hom w x) → g ⋄ f)

{- Note: `is-eqv g` a.k.a. "g is an equivalence" is a
   proposition. This is automatic, as `is-equiv` (the
   analogous property for types) is a proposition. -}

is-idpt+eqv : {x : Ob} (i : Hom x x) → Type j
is-idpt+eqv i = is-idpt i × is-eqv i

{- Note: `is-idpt+eqv i` a.k.a. "being idempotent and an
   equivalence" is still not not a proposition!
   E.g. in the wild semicategory of types and functions,
   the identity on the circle S¹ is an idempotent
   equivalence in ℤ-many ways. -}

{- I define "being a good category" as a property of a
   semicategory. The property states that each object
   comes with an idempotent equivalence. -}

is-good-category = (x : Ob) → Σ (Hom x x) is-idpt+eqv

{- Note: "being a category" a.k.a. "having idempotent
   equivalences" *is* a proposition, but this is not
   trivial. The main goal of the current file is to prove
   this result.

   First, we show that an idempotent equivalence is also a
   standard identity. -}

module idpt+eqv→std {y : Ob} (i : Hom y y) (idpt+eqv : is-idpt+eqv i) where

  idpt = fst idpt+eqv
  eqv = snd idpt+eqv

  {- The idempotent equivalence i is left neutral: -}

  left-neutral : is-left-neutral i
  left-neutral f = w/o-i
    where
      with-i : i ⋄ (i ⋄ f) == i ⋄ f
      with-i =
        i ⋄ (i ⋄ f)
          =⟨ ! ass ⟩
        (i ⋄ i) ⋄ f
          =⟨ ap (λ g → g ⋄ f) idpt ⟩
        i ⋄ f
          =∎
      w/o-i : i ⋄ f == f
      w/o-i = is-equiv.g
                (ap-is-equiv {f = λ g → i ⋄ g} (snd eqv _) (i ⋄ f) f)
                with-i

  {- The proof of right neutrality is completely symmetric
     to the above. Here is the shortened version: -}

  right-neutral : is-right-neutral i
  right-neutral g =
    is-equiv.g
      (ap-is-equiv (fst eqv _) (g ⋄ i) g)
      (ass ∙ ap (λ f → g ⋄ f) idpt)

{- The above shows that an idempotent equivalence is a
   standard equivalence. A "summary statement" will be given
   later.
```

```
   We start the opposite direction with very simple
   observation: Any left-neutral endomorphism is idempotent. -}

left-neutral→idempotent : ∀{y} (f : Hom y y) → is-left-neutral f → is-idpt f
left-neutral→idempotent f l-ntrl = l-ntrl f

{- Of course, the same is true for right-neutral
   endomorphisms, but the above suffices for us.

   We are now ready to prove that a standard identity is an
   idempotent equivalence. -}

module std→idpt+eqv {y : Ob} (i : Hom y y) (std-id : is-standard-id i) where

  l-ntrl = fst std-id
  r-ntrl = snd std-id

  eqv : is-eqv i
  eqv = (λ z → is-eq (λ g → g ⋄ i) (λ h → h) r-ntrl r-ntrl)
        ,
        λ x → is-eq (λ f → i ⋄ f) (λ h → h) l-ntrl l-ntrl

  idpt : is-idpt i
  idpt = left-neutral→idempotent i l-ntrl

{- "Summary statement":
   We now have everything in place to state Lemma TODO (15?) of
   the paper. An endomorphism `i` is an idempotent equivalence
   if and only if it is a standard identity. -}

idpt+eqv⇔std : ∀{y} → (i : Hom y y) → is-idpt+eqv i ⇔ is-standard-id i
idpt+eqv⇔std i = (⇒ , ⇐)
  where
    ⇒ : is-idpt+eqv i → is-standard-id i
    ⇒ p = idpt+eqv→std.left-neutral i p , idpt+eqv→std.right-neutral i p
    ⇐ : is-standard-id i → is-idpt+eqv i
    ⇐ p = std→idpt+eqv.idpt i p , std→idpt+eqv.eqv i p

{- This implies that any two idempotent equivalences are equal. -}

idpt+eqv-unique : ∀{y} → (i₁ i₂ : Hom y y) →
                  is-idpt+eqv i₁ → is-idpt+eqv i₂ → i₁ == i₂
idpt+eqv-unique i₁ i₂ p₁ p₂ =
  i₁
    =⟨ ! (idpt+eqv→std.right-neutral i₂ p₂ i₁) ⟩
  i₁ ⋄ i₂
    =⟨ idpt+eqv→std.left-neutral i₁ p₁ i₂ ⟩
  i₂
    =∎

{- A useful property is 2-out-of-3 for equivalences:
   If we have g ∘ f == h and two out of the three maps
   {f, g, h} are equivalences, then so is the third.
   Here, we only show (and need) one instance, namely
   that it suffices if g and h are equivalences.
   This is easy but a bit tedious. -}

eqv-2-out-of-3 : ∀{w x y} (f : Hom w x) (g : Hom x y)
  → is-eqv g → is-eqv (g ⋄ f) → is-eqv f
eqv-2-out-of-3 {w} {x} {y} f g (p₁ , p₂) (q₁ , q₂) =
  (λ _ → -⋄f-is-eqv)
  ,
  (λ _ → f⋄-is-eqv)
    where

    {- first part -}
    -⋄f : {y : Ob} → Hom x y → Hom w y
    -⋄f = λ h → h ⋄ f
    -⋄g⁻¹ : {z : Ob} → Hom x z → Hom y z
    -⋄g⁻¹ = is-equiv.g (p₁ _)
    -⋄gf : {z : Ob} → Hom y z → Hom w z
    -⋄gf = λ h → h ⋄ (g ⋄ f)
    eq' : ∀{z} → -⋄gf {z} ∘ -⋄g⁻¹ == -⋄f {z}
    eq' {z} = λ= (λ h →
      (-⋄g⁻¹ h) ⋄ (g ⋄ f)
        =⟨ ! ass ⟩
      ((-⋄g⁻¹ h) ⋄ g) ⋄ f
        =⟨ ap (λ k → k ⋄ f) (is-equiv.f-g (p₁ _) _) ⟩
      h ⋄ f
        =∎
      )
    -⋄g⁻¹-equiv : ∀{z} → is-equiv (-⋄g⁻¹ {z})
    -⋄g⁻¹-equiv = is-equiv-inverse (p₁ _)
    -⋄f-is-eqv : ∀{z} → is-equiv (-⋄f {z})
    -⋄f-is-eqv {z} = transport is-equiv eq' (q₁ _ ∘ise -⋄g⁻¹-equiv)

    {- second part -}
```

```
    f⋄- : {v : Ob} → Hom v w → Hom v x
    f⋄- = λ h → f ⋄ h
    g⁻¹⋄- : {v : Ob} → Hom v y → Hom v x
    g⁻¹⋄- = is-equiv.g {f = λ h → g ⋄ h} (p₂ _)
    gf⋄- : {v : Ob} → Hom v w → Hom v y
    gf⋄- = λ h → (g ⋄ f) ⋄ h
    eq : ∀{v} → g⁻¹⋄- {v} ∘ gf⋄- == f⋄- {v}
    eq {v} = λ= (λ h →
      (g⁻¹⋄- ∘ gf⋄-) h
        =⟨ idp ⟩
      g⁻¹⋄- ((g ⋄ f) ⋄ h)
        =⟨ ap g⁻¹⋄- ass ⟩
      g⁻¹⋄- (g ⋄ (f ⋄ h))
        =⟨ is-equiv.g-f (p₂ _) (f ⋄ h)  ⟩
      (f ⋄ h)
        =⟨ idp ⟩
      f⋄- h
        =∎
      )
    g⁻¹⋄-equiv : ∀{v} → is-equiv (g⁻¹⋄- {v})
    g⁻¹⋄-equiv {v} = is-equiv-inverse (p₂ _)
    f⋄-is-eqv : ∀{v} → is-equiv (f⋄- {v})
    f⋄-is-eqv {v} = transport is-equiv eq (g⁻¹⋄-equiv ∘ise q₂ _)

{- Given an equivalence, we can define an idempotent
   equivalence. This construction was already presented in
   "Higher Univalent Categories via Complete Semi-Segal Types"
   (Capriotti-Kraus, POPL'18) and is motivated by work of
   Harpaz and Lurie in higher dimensional category theory. -}

module I {y z} (e : Hom y z) (p : is-eqv e) where

  e⁻¹⋄- : ∀{x} → Hom x z → Hom x y
  e⁻¹⋄- = is-equiv.g (snd p _)

  e⋄- : ∀{x} → Hom x y → Hom x z
  e⋄- = _⋄_ e

  I : Hom y y
  I = e⁻¹⋄- e

  {- In comments, we write I(e) for the I constructed above
     (e is a module parameter).

     It is easy to see that I(e) is right neutral for e: -}

  e⋄I : e ⋄ I == e
  e⋄I = is-equiv.f-g (snd p _) e

  {- Also easy (but more work):
     I is left neutral in general. -}

  l-ntrl : is-left-neutral I
  l-ntrl f =
    I ⋄ f
      =⟨ ! (is-equiv.g-f (snd p _) _) ⟩
    e⁻¹⋄- (e⋄- (I ⋄ f))
      =⟨ ap e⁻¹⋄- (! ass) ⟩
    e⁻¹⋄- ((e ⋄ I) ⋄ f)
      =⟨ ap (λ g → e⁻¹⋄- (g ⋄ f)) e⋄I ⟩
    e⁻¹⋄- (e ⋄ f)
      =⟨ is-equiv.g-f (snd p _) _ ⟩
    f
      =∎

  {- We do not need the other analogous properties. This
     is enough to see that I is an idempotent equivalence. -}

  I-is-idpt+eqv : is-idpt+eqv I
  I-is-idpt+eqv = left-neutral→idempotent
    I l-ntrl ,
    eqv-2-out-of-3 I e p (transport is-eqv (! e⋄I) p)

{- If an endomorphism e is an equivalence, then it is
   idempotent if and only if it is equal to I(e); and
   this connection even forms an equivalence of types
-}

module e-vs-I {y : Ob} (e : Hom y y) (p : is-eqv e) where
  open I

  e-I-idpt : (e == I e p) ≃ is-idpt e
  e-I-idpt =
    e == I e p
      ≃⟨ ap-equiv (e⋄- e p , snd p _) e (I e p) ⟩
    e ⋄ e == e ⋄ I e p
      ≃⟨ transport (λ expr →
```

```
           (e ⋄ e == e ⋄ I e p) ≃ (e ⋄ e == expr)) (e⋄I e p) (ide _) ⟩
      e ⋄ e == e
        ≃⟨ ide _ ⟩
      is-idpt e
        ≃∎

  {- Finally, put all the previous lemmas together to show that
     the type of idempotent equivalences is a proposition. We do
     this by showing that, if we assume that this type has one
     element i₀, then i₀ is the only element. -}

  module unique (y : Ob) (i₀ : Hom y y) (idpt+eqv₀ : is-idpt+eqv i₀) where

    open I
    open e-vs-I

    idpt₀ = fst idpt+eqv₀
    eqv₀ = snd idpt+eqv₀

    {- Using a chain of equivalences, we show that the type
       of idempotent equivalences is trivial (given one
       single idempotent equivalence, see the module
       parameter). -}

    idpt-to-=i₀ =
      Σ (Hom y y) (λ i → is-idpt+eqv i)
        ≃⟨ ide {i = j} _ ⟩
      Σ (Hom y y) (λ i → is-idpt i × is-eqv i)
        ≃⟨ Σ-emap-r (λ i → ×-comm) ⟩
      Σ (Hom y y) (λ i → Σ (is-eqv i) λ eqv → (is-idpt i))
        ≃⟨ Σ-emap-r (λ i → Σ-emap-r λ eqv → e-I-idpt i eqv ⁻¹) ⟩
      Σ (Hom y y) (λ i → Σ (is-eqv i) λ eqv → i == I i eqv)
        ≃⟨ Σ-emap-r (λ i → Σ-emap-r λ p → coe-equiv
             (ap (λ i' → (i == i')) (
                   idpt+eqv-unique (I i p) i₀ (I-is-idpt+eqv i p) idpt+eqv₀))) ⟩
      Σ (Hom y y) (λ i → Σ (is-eqv i) λ _ → i == i₀)
        ≃⟨ Σ-emap-r (λ i → ×-comm) ⟩
      Σ (Hom y y) (λ i → Σ (i == i₀) λ _ → (is-eqv i))
        ≃⟨ Σ-assoc ⁻¹ ⟩
      (Σ (Σ (Hom y y) (λ i → (i == i₀))) λ ip → (is-eqv (fst ip)))
        ≃⟨ Σ-emap-l {A = Unit}
                    {B = Σ (Hom y y) λ i → (i == i₀)}
                    (λ ip → (is-eqv (fst ip)))
                    (contr-equiv-Unit (pathto-is-contr i₀) ⁻¹) ⁻¹ ⟩
      (Σ Unit λ _ → is-eqv i₀)
        ≃⟨ Σ₁-Unit ⟩
      is-eqv i₀
        ≃⟨ contr-equiv-Unit (inhab-prop-is-contr eqv₀) ⟩
      Unit
        ≃∎

    {- In other words, the type of idempotent equivalences
       is contractible. -}

    unique-idpt+eqv : is-contr (Σ (Hom y y) (λ i → is-idpt+eqv i))
    unique-idpt+eqv = equiv-preserves-level (idpt-to-=i₀ ⁻¹)


{-                     THEOREMS
                       ========

   With the help of the above lemmas, we show our main
   results:

   (1) A semicategory is a good category if and only if
       it is a standard category.

   (2) "Being a good category" is a propositional (proof-
       irrelevant) property of a semicategory.
-}

good-iff-standard : ∀ {j₁ j₂} (C : SemiCategory j₁ j₂) →
                    is-good-category C ⇔ is-standard-category C
good-iff-standard C = ⇒ , ⇐
  where
  ⇒ : is-good-category C → is-standard-category C
  ⇒ igc x = fst (igc x) , fst (idpt+eqv⇔std C _) (snd (igc x))
  ⇐ : is-standard-category C → is-good-category C
  ⇐ isc x = fst (isc x) , snd (idpt+eqv⇔std C _) (snd (isc x))

goodness-is-prop : ∀ {j₁ j₂} (C : SemiCategory j₁ j₂) → is-prop (is-good-category C)
goodness-is-prop C = inhab-to-contr-is-prop λ idpt+eqvs →
  WeakFunext.weak-λ= λ y → unique.unique-idpt+eqv C y (fst(idpt+eqvs y)) (snd(idpt+eqvs y))
```