Functors

```
module FunctorP where

Functor : ∀ 𝒰 → 𝒰 ⁺ ̇
Functor 𝒰 = Σ Fn : (𝒰 ̇ → 𝒰 ̇) , Σ Fm : (∀{X Y} → (f : X → Y) → Fn X → Fn Y) , (∀{X Y
Z} → (f : X → Y) → (g : Z → X) → ∀ x → (Fm f) (Fm g x) = Fm (f ∘ g) x) × (∀{X} → Fm
id ~ id {X = Fn X})

module Functor (func : Functor 𝒰) where

 Fn : 𝒰 ̇ → 𝒰 ̇
 Fn = func .pr₁

 Fm : (∀{X Y} → (f : X → Y) → Fn X → Fn Y)
 Fm = func .pr₂ .pr₁

 Fm-comp : (∀{X Y Z} → (f : X → Y) → (g : Z → X) → ∀ x → (Fm f) (Fm g x) = Fm (f ∘
g) x)
 Fm-comp = func .pr₂ .pr₂ .pr₁

 Fm-id : ∀{X} → Fm id ~ id {X = Fn X}
 Fm-id = func .pr₂ .pr₂ .pr₂
```