HOTT EXERCISES IN AGDA

ERIC BAILEY

Contents

1. Chapter 1

1.1. **Exercise 1.**

Use the same options as HoTT-Agda.

```
{-# OPTIONS --without-K --rewriting #-}
```

Declare the module.

```
module HoTT.Chapter1.Ex1 where
```

Import the propositional equality module for ≡.

```
open import Relation.Binary.PropositionalEquality
```

Given functions $f : A \to B$ and $g : B \to C$, define their **composite** $g \circ f : A \to C$.

```
_∘_ : ∀ {A B C : Set} → (B → C) → (A → B) → A → C
g ∘ f =  x → g (f x)
```

Prove that function composition is associative, i.e.

$$h \circ (g \circ f) \equiv (h \circ g) \circ f$$

```
∘-assoc : ∀ {A B C D : Set} {f : A → B} {g : B → C} {h : C → D} →
          h ∘ (g ∘ f) ≡ (h ∘ g) ∘ f
∘-assoc = refl
```