```
{-# OPTIONS --guardedness #-}

open import Codata.Musical.Notation
open import Data.Nat using (ℕ; suc; zero)
open import Relation.Binary.Core using (Rel)
open import Relation.Binary.Bundles using (Setoid)
open import Relation.Binary.Definitions using (Reflexive; Symmetric; Transitive)
open import Relation.Binary.PropositionalEquality using (_≡_; subst; subst₂) renaming (sym to eqSym; trans
import Level using (zero)
open import Data.Maybe using (Maybe; nothing; just)
open import Data.Maybe.Properties
open import Data.Bool using (Bool; true; false)
open import Data.Product
open import Data.Sum
open import Function.Base using (case_of_)
open import Relation.Nullary using (contradiction)

Id : Set
Id = ℕ

Val : Set
Val = ℕ

State : Set
State = Id → Val

record Trace₃ : Set where
         coinductive
         constructor mkTr
         field
           hd : State
           tl : Maybe Trace₃


open Trace₃

record _≈_ (tr₁ tr₂ : Trace₃) : Set where
             coinductive
             field
               hd : hd tr₁ ≡ hd tr₂
               tl : (tl tr₁ ≡ nothing × tl tr₂ ≡ nothing)
                    ⊔
                    ∃ {A = (Trace₃ × Trace₃)} λ x → (
                        tl tr₁ ≡ just (proj₁ x)
                        ×
                        tl tr₂ ≡ just (proj₂ x)
                        ×
                        (proj₁ x) ≈ (proj₂ x))
```