```
{-# 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


data Trace₁ : Set where
             tnil : State → Trace₁
             tcons : State → ∞ Trace₁ → Trace₁


data _≈_ : Rel Trace₁ Level.zero where
              tnil : ∀ {st} → (tnil st) ≈ (tnil st)
              tcons : ∀ {e tr₁ tr₂} → ∞ (♭ tr₁ ≈ ♭ tr₂) → (tcons e tr₁) ≈ (tcons e tr₂)
```