```
nominal sorts Config, Actor
moninal preds locality, ask, choose, travelTo, path : Config Actor Actor

rigid sorts ActorType
rigid ops User, Bike, Dock, Region : → ActorType

flexible op type : → ActorType

∀ k : Config; a, b : Actor · locality(k, a, b)
  ⇒ @ k : (
       (@ a : type = User ∧ @ b : type = Bike)
    ∨ (@ a : type = User ∧ @ b : type = Region)
    ∨ (@ a : type = Bike ∧ @ b = Dock)
    ∨ (@ a : type = Bike ∧ @ b = Region)
  )

...

flexible pred travelling

∀ k : Config; u : Actor
· @ (k, u) : travelling
  ⇔ (
    @ u : type = User
    ∧ ∃ r : Actor · @ r : type = Region ∧ travelTo(k, u, r)
  )

...

nominal pred Take : Config Config

∀ k : Config; u, b, d, r : Actor
· @ k : (@ u : type = User ∧ @ b : type = Bike ∧ @ d : type = Dock ∧ @ r : type = Region)
  ∧ locality(k, u, r) ∧ ask(k, u, r) ∧ locality(k, b, d) ∧ locality(k, d, r) ∧ choose(k, u, b)
  ⇒
  ∀ k' : Config
  · Take(k, k')
    ⇒
    locality(k', u, b) ∧ locality(k', b, r)
    ∧ ∃ r' : Actor · @ r' : type = Region ∧  travelTo(k', u, r')

∀ k : Config; u, b, d, r : Actor
· @ k : (@ u : type = User ∧ @ b : type = Bike ∧ @ d : type = Dock ∧ @ r : type = Region)
  ∧ locality(k, u, r) ∧ ask(k, u, r) ∧ locality(k, b, d) ∧ locality(k, d, r) ∧ choose(k, u, b)
  ⇒
  [ Take ] store k' : Config ·
    locality(k', u, b) ∧ locality(k', b, r)
    ∧ ∃ r' : Actor · @ r' : type = Region ∧  travelTo(k', u, r')
```