```
-- disable the K axiom:

{-# OPTIONS --without-K #-}


-- Agda version 2.5.2


module setoid-cwf-pt1 where

open import basic-setoids
open import dependent-setoids
open import subsetoids
open import iterative-sets
open import iterative-sets-pt2
open import iterative-sets-pt3

ctx : Set1
ctx = V

ctx-maps : (a b : ctx) -> setoid
ctx-maps a b = (κ a => κ b)

subst : (a b : ctx) -> Set
subst a b = || ctx-maps a b ||

idsubst : {Γ : ctx} -> || κ Γ => κ Γ ||
idsubst {Γ} = idmap {κ Γ}

Ty : (Γ : ctx) -> classoid
Ty Γ =  setoidmaps1 (κ Γ) (subsetoids VV)

ty : (Γ : ctx) -> Set1
ty Γ =  ||| Ty Γ |||

Raw : (Γ : ctx) -> classoid
Raw Γ = setoidmaps1 (κ Γ) VV

raw : (Γ : ctx) -> Set1
raw Γ = ||| Raw Γ |||

record tm (Γ : ctx) (A : ty Γ) : Set1 where
  field
    trm : raw Γ
    corr : (x : || κ Γ ||) -> member (trm • x)  (A • x)

-- Define interpretation of the type-theoretic judgements
--  Γ context
--  Γ ==> A type
--  Γ ==> A == B
--  Γ ==> a :: A
--  Γ ==> a == b :: A

infix 10  _==>_==_
```

```
_==>_==_ : (Γ : ctx) -> (A B : ty Γ) -> Set
Γ ==> A == B = exteq1 A B

infix 10  _==>_::_

_==>_::_ : (Γ : ctx) -> (a : raw Γ) -> (A : ty Γ) -> Set
Γ ==> a :: A =  (x : || κ Γ ||) -> member (a • x)  (A • x)

infix 10  _==>_==_::_

_==>_==_::_ : (Γ : ctx) -> (a b : raw Γ) -> (A : ty Γ) -> Set
Γ ==> a == b :: A = and ((x : || κ Γ ||) -> (a • x) ≐ (b • x))
                           (and (Γ ==> a :: A)  (Γ ==> b :: A))

tyrefl :  (Γ : ctx)
--
         -> (A : ty Γ)
-- ------------------------
         -> Γ ==> A == A
--
tyrefl Γ A = λ x →  refl' (subsetoids VV) (A • x)

tysym :  (Γ : ctx) -> (A B : ty Γ)
--
         -> Γ ==> A == B
--   --------------------------
         -> Γ ==> B == A
--
tysym Γ A B p = λ x → sym' {subsetoids VV} {A • x} {B • x} (p x)

tytra :  (Γ : ctx) -> (A B C : ty Γ)
--
         -> Γ ==> A == B   -> Γ ==> B == C
--  -------------------------------------
                -> Γ ==> A == C
--
tytra Γ A B C p q = λ x → tra' {subsetoids VV} {A • x} {B • x} {C • x}  (p x) (q
x)


tmrefl :  (Γ : ctx) -> (A : ty Γ) -> (a : raw Γ)
--
         -> Γ ==> a :: A
--    ------------------------
         -> Γ ==> a == a :: A
--
tmrefl Γ A a p = pair (λ x → refV _) (pair p p)

tmsym :  (Γ : ctx) -> (A : ty Γ) -> (a b : raw Γ)
--
         -> Γ ==> a == b :: A
--    ------------------------
         -> Γ ==> b == a :: A
--
```

```
tmsym Γ A a b p = pair (λ x → symV (prj1 p x)) (pair (prj2 (prj2 p)) (prj1 (prj2
p)))

tmtra :  (Γ : ctx) -> (A : ty Γ) -> (a b c : raw Γ)
--
         -> Γ ==> a == b :: A    -> Γ ==> b == c :: A
--  --------------------------------------------------
         -> Γ ==> a == c :: A
--
tmtra Γ A a b c p q = pair (λ x → traV (prj1 p x) (prj1 q x)) (pair (prj1 (prj2
p)) (prj2 (prj2 q)))



elttyeq :  (Γ : ctx) ->  (a : raw Γ)  -> (A B : ty Γ)
--
      -> Γ ==> a :: A      -> Γ ==> A == B
--  --------------------------------------------
           -> Γ ==> a :: B
--
elttyeq Γ a A B p q = λ x → eq-lemma VV (A • x) (B • x) (q x) (a • x) (p x)

elteqtyeq :  (Γ : ctx) ->  (a b : raw Γ)  -> (A B : ty Γ)
--
      -> Γ ==>  a == b :: A     -> Γ ==> A == B
--  ---------------------------------------------
           -> Γ ==>  a == b :: B
--
elteqtyeq Γ a b A B p q = pair (prj1 p)
                              (pair (elttyeq Γ a A B (prj1 (prj2 p)) q)
                                    (elttyeq Γ b A B (prj2 (prj2 p)) q))

-- substitutions


sub : {Δ Γ : ctx}  -> raw Γ ->  subst Δ Γ -> raw Δ
sub a f =  comp1 a f

Sub : {Δ Γ : ctx}  -> ty Γ -> subst Δ Γ -> ty Δ
Sub A f =  comp1 A f

sub-id-prop : {Γ : ctx}  -> (a : raw Γ) ->
       << Raw Γ >> sub {Γ} {Γ} a (idsubst {Γ}) ~ a
sub-id-prop {Γ} a = λ x → refl' VV _

sub-comp-prop : {Θ Δ Γ : ctx}  -> (a : raw Γ)
       ->  (f : subst Δ Γ) -> (g : subst Θ Δ) ->
       << Raw Θ >> sub {Θ} {Γ} a (f ° g)  ~ sub {Θ} {Δ} (sub {Δ} {Γ} a f) g
sub-comp-prop a f g = λ x → refl' VV _

Sub-id-prop : {Γ : ctx}  -> (A : ty Γ) ->
       << Ty Γ >> Sub {Γ} {Γ} A (idsubst {Γ}) ~ A
Sub-id-prop {Γ} a = λ x → refl' (subsetoids VV) (a • x)

Sub-comp-prop : {Θ Δ Γ : ctx}  -> (A : ty Γ)
```

```
       ->  (f : subst Δ Γ) -> (g : subst Θ Δ) ->
       << Ty Θ >> Sub {Θ} {Γ} A (f ° g)  ~ Sub {Θ} {Δ} (Sub {Δ} {Γ} A f) g
Sub-comp-prop {Θ} {Δ} {Γ} A f g = λ x → refl' (subsetoids VV) (Sub {Θ} {Δ} (Sub
{Δ} {Γ} A f) g • x)



tyeq-subst :  (Δ Γ : ctx) -> (A B : ty Γ) -> (f : subst Δ Γ)
--
                  -> Γ ==> A == B
--      ------------------------------------------------
         -> Δ ==> (Sub {Δ} {Γ} A f) == (Sub {Δ} {Γ} B f)
--
tyeq-subst Δ Γ A B f p = λ x → p (ap f x)


elt-subst :  (Δ Γ : ctx) -> (a : raw Γ) -> (A : ty Γ) -> (f : subst Δ Γ)
--
          -> Γ ==> a :: A
--   --------------------------------------------------------
         -> Δ ==> (sub {Δ} {Γ} a f) :: (Sub {Δ} {Γ} A f)
--
elt-subst Δ Γ a A f p = λ x → p (ap f x)



elteq-subst :  (Δ Γ : ctx) -> (a b : raw Γ) -> (A : ty Γ) -> (f : subst Δ Γ)
--
          -> Γ ==> a == b :: A
--   --------------------------------------------------------------------------
         -> Δ ==> (sub {Δ} {Γ} a f) == (sub {Δ} {Γ} b f) :: (Sub {Δ} {Γ} A f)
--

elteq-subst Δ Γ a b A f p = pair (\x -> prj1 p (ap f x))
                                 (pair (elt-subst Δ Γ a A f (prj1 (prj2 p)))
                                       (elt-subst Δ Γ b A f (prj2 (prj2 p))))




-- context extensions

-- we use

-- Sub-to-V : (S : subsetoid VV) -> V

-- ext-Sub-to-V :  (S T : subsetoid VV) -> equal-subsetoids S T -> (Sub-to-V S) ≐
(Sub-to-V T)


subsetoids-VV :  setoidmap11 (subsetoids VV) VV
subsetoids-VV = record { op = Sub-to-V
                       ; ext = ext-Sub-to-V }
```

```
Ext : (Γ : ctx) -> (A : ty Γ) -> ctx
Ext Γ A = sigmaV Γ (comp01 subsetoids-VV A)

infixl 20 _▷_

_▷_ : (Γ : ctx) -> (A : ty Γ) -> ctx
Γ ▷ A = Ext Γ A

pExt : (Γ : ctx) -> (A : ty Γ) -> subst (Γ ▷ A) Γ
pExt Γ A = (π1 (κ Γ) (κ° (comp01 subsetoids-VV A))) ° (κ-sigmaV-fwd Γ (comp01
subsetoids-VV A))

vExt : (Γ : ctx) -> (A : ty Γ) -> raw (Γ ▷ A)
vExt Γ A = pj2-sigmaV  Γ (comp01 subsetoids-VV A)

asm : (Γ : ctx)
--
      -> (A : ty Γ)
--   -----------------------------------------------------
      -> Γ ▷ A ==> vExt Γ A :: (Sub {Γ ▷ A} {Γ} A (pExt Γ A))
--
asm Γ A u = let hyp : vExt Γ A • u ∊
                       comp01 subsetoids-VV A • pj1 u
                hyp = pj2-sigmaV-prop Γ  (comp01 subsetoids-VV A) u
                hyp' : member (vExt Γ A • u) (V-to-Sub (comp01 subsetoids-VV A •
pj1 u))
                hyp' = membV-member (vExt Γ A • u) (comp01 subsetoids-VV A • pj1
u) hyp
                main : member (vExt Γ A • u) (Sub {Γ ▷ A} {Γ} A (pExt Γ A) • u)
                main = hyp'
            in main



ext-op : (Δ Γ : ctx) -> (A : ty Γ)
        ->  (f : subst Δ Γ) -> (a : raw Δ)
        -> Δ ==> a :: (Sub {Δ} {Γ} A f)
        -> || κ Δ || -> || κ (Γ ▷ A) ||
ext-op Δ Γ A f a p u = (ap f u) , (pj1 (p u))

ext-ext : (Δ Γ : ctx) -> (A : ty Γ)
        ->  (f : subst Δ Γ) -> (a : raw Δ)
        -> (p : Δ ==> a :: (Sub {Δ} {Γ} A f))
        -> (u  v : || κ Δ ||)
        -> < κ Δ > u ~ v
        -> < κ (Γ ▷ A) > ext-op Δ Γ A f a p u ~ ext-op Δ Γ A f a p v
ext-ext Δ Γ A f a p u v q =
       let  lm1 : < κ Γ > ap f u ~ ap f v
            lm1 = (extensionality f _ _ q)
            lmA : << subsetoids VV >> A • ap f u ~ (A • ap f v)
            lmA  = extensionality1 A (ap f u) (ap f v) lm1
```

```
            lmB : << VV >> (comp01 subsetoids-VV A • ap f u) ~ (comp01 subsetoids-
VV A • ap f v)
            lmB = extensionality11 subsetoids-VV (A • ap f u) (A • ap f v) lmA
            lmC : << VV >> ap1 (ι (Sub {Δ} {Γ} A f • u)) (pj1 (p u)) ~ (a • u)
            lmC = pj2 (p u)
            lmD : << VV >> ap1 (ι (Sub {Δ} {Γ} A f • v)) (pj1 (p v)) ~ (a • v)
            lmD = pj2 (p v)
            lmE : << VV >> (a • u) ~ (a • v)
            lmE = extensionality1 a u v q
            lm2 :  (comp01 subsetoids-VV A • ap f u) ‣ pj1 (p u) ≐
                   (comp01 subsetoids-VV A • ap f v) ‣ pj1 (p v)
            lm2 = traV lmC (traV lmE (symV lmD))

       in pairV-ext lm1 lm2

ext : (Δ Γ : ctx) -> (A : ty Γ)
      ->  (f : subst Δ Γ) -> (a : raw Δ)
      -> Δ ==> a :: (Sub {Δ} {Γ} A f)
      -> subst Δ (Γ ▷ A)
ext Δ Γ A f a p = record { op = ext-op Δ Γ A f a p
                         ; ext = ext-ext Δ Γ A f a p }

ext-prop1 : (Δ Γ : ctx) -> (A : ty Γ)
      ->  (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: (Sub {Δ} {Γ} A f))
      -> < κ Δ => κ Γ >  (pExt Γ A) ° (ext Δ Γ A f a p) ~ f
ext-prop1  Δ Γ A f a p = λ x → refl (κ Γ) (ap f x)

ext-prop2 : (Δ Γ : ctx) -> (A : ty Γ)
      ->  (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: (Sub {Δ} {Γ} A f))
      -> << Raw Δ >>  (sub {Δ} {Γ ▷ A} (vExt Γ A) (ext Δ Γ A f a p)) ~ a
ext-prop2  Δ Γ A f a p x = let lm5 : ((comp01 subsetoids-VV A) • (ap f x)) ‣ (pj1
(p x)) ≐ (a • x)
                               lm5 = pj2 (p x)
                               main : << VV >> sub  {Δ} {Γ ▷ A} (vExt Γ A) (ext Δ
Γ A f a p) • x ~ (a • x)
                               main = lm5
                           in main

ext-prop3 : (Γ : ctx) -> (A : ty Γ) ->
            < κ (Γ ▷ A) => κ (Ext Γ A) >  (ext (Γ ▷ A) Γ A (pExt Γ A) (vExt Γ A)
(asm Γ A)) ~ idsubst {Γ ▷ A}
ext-prop3 Γ A (x , y) = let    lm2 :  < κ (Γ ▷ A) >
                                        (x , y)  ~  (x , y)
                               lm2 = refl (κ (Γ ▷ A)) (x , y)
                               lm1 :  < κ (Γ ▷ A) >
                                        (ap (pExt Γ A) (x , y)  ,  (pj1 ((asm Γ A)
(x , y))))
                                       ~  (x , y)
                               lm1 = lm2
                               main : < κ (Γ ▷ A) >
                                       (ext-op (Γ ▷ A) Γ A (pExt Γ A) (vExt Γ A)
```

```
(asm Γ A)) (x , y)
                                        ~  (x , y)
                               main = lm1
                        in main

ext-prop4-lm : (Θ Δ Γ : ctx) -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: (Sub {Δ} {Γ} A f))
      -> (h : subst Θ Δ)
      -> Θ ==> sub {Θ} {Δ} a h :: Sub {Θ} {Γ} A (f ° h)
ext-prop4-lm  Θ Δ Γ A f a p h =
          let lm1 :  Θ ==> sub {Θ} {Δ} a h :: Sub {Θ} {Δ} (Sub {Δ} {Γ} A f) h
              lm1  = elt-subst Θ Δ a (Sub {Δ} {Γ} A f) h p
              lm2 : Θ ==> Sub {Θ} {Δ} (Sub {Δ} {Γ} A f) h == Sub {Θ} {Γ} A (f ° h)
              lm2 = tysym Θ (Sub {Θ} {Γ} A (f ° h)) (Sub {Θ} {Δ} (Sub {Δ} {Γ} A f)
h)
                     (λ x → (Sub-comp-prop {Θ} {Δ} {Γ} A f h x))
              main :  Θ ==> sub {Θ} {Δ} a h :: Sub {Θ} {Γ} A (f ° h)
              main = elttyeq Θ (sub {Θ} {Δ} a h) (Sub {Θ} {Δ} (Sub {Δ} {Γ} A f) h)
(Sub {Θ} {Γ} A (f ° h)) lm1 lm2
          in main


ext-prop4 : (Θ Δ Γ : ctx) -> (A : ty Γ)
      -> (f : subst Δ Γ) -> (a : raw Δ)
      -> (p : Δ ==> a :: (Sub {Δ} {Γ} A f))
      -> (h : subst Θ Δ)
      -> < κ Θ => κ (Γ ▷ A) > (ext Δ Γ A f a p) ° h ~ (ext Θ Γ A (f ° h) (sub {Θ}
{Δ} a h) (ext-prop4-lm  Θ Δ Γ A f a p h ))
ext-prop4 Θ Δ Γ A f a p h u =
              let lm2a :  Γ ‣ ap f (ap h u) ≐ Γ ‣ ap (f ° h) u
                  lm2a = refV _
                  lm1 :  < κ (Γ ▷ A) >  ((ap f (ap h u)) ,  (pj1 (p (ap h
u))))      ~
                                          ((ap (f ° h) u) , (pj1 ((ext-prop4-lm Θ
Δ Γ A f a p h) u)))
                  lm1 = pairV-ext lm2a (refV _)
                  main : < κ (Γ ▷ A) >  ap (ext Δ Γ A f a p)  (ap h u) ~
                                        ap (ext Θ Γ A (f ° h) (sub {Θ} {Δ} a h)
(ext-prop4-lm Θ Δ Γ A f a p h)) u
                  main = lm1
              in main
```