Множества, Сетоиды, Теорема Диаконеску

Set — не множество

```
Theorem choice :
 forall (A B : Type) (R : A->B->Prop),
   (forall x : A, exists y : B, R x y) ->
    exists f : A->B, (forall x : A, R x (f x)).
```

Сетоид

Определение

Сетоид — множество с отношением эквивалентности

```
Reflx: {A : Type} -> (R: A -> A -> Type) -> Type
Reflx {A} R = (x : A) -> R x x

Symm: {A : Type} -> (R: A -> A -> Type) -> Type
Symm {A} R = (x : A) -> (y : A) -> R x y -> R y x

Trans: {A : Type} -> (R: A -> A -> Type) -> Type
Trans {A} R = (x : A) -> (y : A) -> (z : A) -> R x y -> R y z -> R x z

data IsEquivalence: {A : Type} -> (R: A -> A -> Type) -> Type where
    EqProof: {A: Type} -> (R: A -> A -> Type) ->
       Reflx {A} R -> Symm {A} R -> Trans {A} R -> IsEquivalence {A} R

record Setoid where
    constructor MkSetoid
    Carrier: Type
    Equiv: Carrier -> Carrier -> Type
    EquivProof: IsEquivalence Equiv
```

```
data Map: (A:Setoid) -> (B:Setoid) -> Type where
  MkMap: {A:Setoid} -> {B:Setoid} -> (f: (Carrier A) -> (Carrier B)) ->
     ({x:Carrier A} -> {y:Carrier A} ->
     ((Equiv A) x y) -> ((Equiv B) (f x) (f y))) -> Map A B

MapF: {A:Setoid} -> {B:Setoid} -> Map A B -> (Carrier A -> Carrier B)
MapF (MkMap {A} {B} f ext) = f

MapExt: {A:Setoid} -> {B:Setoid} -> (p: Map A B) ->
     ({x:Carrier A} -> {y:Carrier A} -> ((Equiv A) x y) -> ((Equiv B) (MapF p x) (MapF p y
MapExt (MkMap {A} {B} f ext) = ext

Rel: Type -> Type -> Type
Rel a b = a -> b -> Type

postulate ext_ac: {I: Setoid} -> {S: Setoid} ->
  (A: Rel (Carrier I) (Carrier S)) ->
  ((x: Carrier I) -> (g : Carrier S ** A x g)) ->
  (chs: (Map I S) ** ((w: Carrier I) -> A w ((MapF chs) w)))

excluded_middle: (P: Type) -> Or P (Not P)
```

Аксиома выбора в HoTT