```
{- how to go beyond epsilon0 with one universe -}

nat = data {$0,$S (x:nat)} : Set,

fun (A:Set,B:Set) = (x:A) -> B : Set,

exp (A:Set) = fun A A : Set,

iterate (A:Set,f:exp A,n:nat) =
 case n of
  {$0 -> \ x -> x,
   $S n1 -> \ x -> f (iterate A f n1 x)
  } : exp A,

prime (n:nat,A:Set) =
 case n of
   {$0 -> A,
    $S n1 -> exp (prime n1 A)
   } : Set,

T (A:Set) = fun (fun (fun nat A) A) (exp (exp A)) : Set,

{- limit structures -}

limstruct (A:Set) =
 sig {O:A,S:fun A A,L:fun (fun nat A) A} : Set,

limT (A:Set) =
 struct
   {O = \ L f x -> x,
    S = \ u L f x -> f (u L f x),
    L = \ F L f x -> L (\ n -> F n L f x)
   } : limstruct (T A),

Pi (G:(n:nat) -> Set) = (n:nat) -> G n : Set,

limPi (G: (n:nat) -> Set,limG:(n:nat) -> limstruct (G n)) =
 struct
   {O = \ n -> (limG n).O,
    S = \ u n -> (limG n).S (u n),
    L = \ F n -> (limG n).L (\ k -> F k n)
   }
     : limstruct (Pi G),


{- an example of a limit structure; the Hardy hierarchy -}

Hardy =
 struct {O = \ x -> $S x,
         S = \ f x -> iterate nat f x x,
         L = \ F x -> F x x
        }
   : limstruct (fun nat nat),

{- predicative types of ordinals -}

Ord = (A:Set) -> T A : Type,

Zero = \ A L f x -> x : Ord,

Succ (u:Ord) = \ A L f x -> f (u A L f x) : Ord,

Lim (F:(n:nat) -> Ord) = \ A L f x -> L (\ n -> F n A L f x) : Ord,

{- the ordinal omega -}

omega = \ A L f x -> L (\ n -> iterate A f n x) : Ord,

{- exponentiation: u |--> w^u -}
```

```
power (A:Set) =
 \ u L -> u (\ F x -> L (\ n -> F n x)) (omega A L)
 : fun (T (exp A)) (T A),

{- how to get beta |--> epsilon_beta -}

Phi (A:Set) = Pi (\ n -> T (prime n A)) : Set,

limPhi (A:Set) = limPi (\n -> T (prime n A)) (\ n -> limT (prime n A))
 : limstruct (Phi A),

powerPhi (A:Set) =
    \ u n -> power (prime n A) (u ($S n))
 : fun (Phi A) (Phi A),

ZeroPhi (A:Set) = (limPhi A).O : Phi A,

SuccPhi (A:Set) = (limPhi A).S : fun (Phi A) (Phi A),

LimPhi (A:Set) = (limPhi A).L : fun (fun nat (Phi A)) (Phi A),

nextPhi (A:Set) =
 \ u -> LimPhi A (\ n -> iterate (Phi A) (powerPhi A) n (SuccPhi A u))
 : fun (Phi A) (Phi A),

epsilon (A:Set) =
 \ u -> u (LimPhi A) (nextPhi A) (nextPhi A (ZeroPhi A)) $0
 : fun (T (Phi A)) (T A),

{- we can define for instance epsilon_omega -}

example (A:Set) = epsilon A (omega (Phi A)) : T A,

fast_epsilon_omega =
 example (fun nat nat) (Hardy.L) (Hardy.S) (Hardy.O)
 : fun nat nat
```