controllable-refinement

Version 7.9

Lîm Tsú-thuàn

November 13, 2020

The type system based on STLC, introducing user-controllable type refinement. Meta type variables are A, B. Meta predicate variable is P. A type with predicate write as A_P, introduce predicate write as P_+, eliminate write as P_-, $P_{?+}$ for anyway introduce, $P_{?-}$ for anyway eliminate.

- verify-predicate

$$\frac{\Gamma, f : A_P \to B, x : A_P}{fx : B}$$

- introduce-predicate

$$\frac{\Gamma, f : (p : A_{P_+}) \to B, x : A}{fx : B, x : A_P, p : A}$$

- eliminate-predicate

$$\frac{\Gamma, f : (p : A_{P_-}) \to B, x : A_P}{fx : B, x : A, p : A_P}$$

- anyway-introduce-predicate

$$\frac{\Gamma, f : A_{P_{?+}} \to B, x : A}{fx : B, x : A_P} \quad \frac{\Gamma, f : A_{P_{?+}} \to B, x : A_P}{fx : B, x : A_P}$$

- anyway-eliminate-predicate

$$\frac{\Gamma, f : A_{P_{?-}} \to B, x : A_P}{fx : B, x : A} \quad \frac{\Gamma, f : A_{P_{?-}} \to B, x : A}{fx : B, x : A}$$

Notice that it can be extended to with polymorphism without changing previous definition.

1 extension: related predicate

Sometime we would like to update related predicate, for example:

```
(: open (-> (file {writable readable})))
(: write (-> (file {writable}) void))
(: close (-> (file {+close}) void))

(let ([file (open "xxx")])
  (write file "hello")
  (close file)
  (write file "hello"))
```

where `+close` also means `?-readable ?-writable`. This extension is quite simple, we can add related predicate information into environment, rewrite predicates of type while parsing type.