非手続き型言語13回目課題 解答例

June 8, 2020

1　MLOO 問 3.5 の 3

問題: fun A x y z = z y x の型
　A: τ_1, x: τ_2, y : τ_3, z : τ_4, z y x : τ_5 とおく．求めたいのは

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_5$$

関数 z について

$$\tau_4 = \tau_3 \to \tau_2 \to \tau_5$$

τ_4 に関する式を τ_1 の中にいれて

$$\tau_1 = \tau_2 \to \tau_3 \to (\tau_3 \to \tau_2 \to \tau_5) \to \tau_5$$

これ以上簡単にならないので τ_2 を'a, τ_3 を'b, τ_5 を'c としてこの関数の型は

```
'a -> 'b -> ('b -> 'a -> 'c) -> 'c
```

2　MLOO 問 3.5 の 4

問題: fun B f g = f g g の型
　B: τ_1, f: τ_2, g : τ_3, f g g : τ_4 とおく．求めたいのは

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4$$

関数 f について

$$\tau_2 = \tau_3 \to \tau_3 \to \tau_4$$

τ_2 に関する式を τ_1 の式に代入して

$$\tau_1 = (\tau_3 \to \tau_3 \to \tau_4) \to \tau_3 \to \tau_4$$

これ以上簡単にならないので τ_3 を'a, τ_4 を'b としてこの関数の型は

```
('a -> 'a -> 'b) -> 'a -> 'b
```

3 MLOO 問 3.7 の 2,3,4,5

準備:

(i) fun f x y z = x y z
スライドで解説したように
f: τ_1, x : τ_2, y: τ_3, z: τ_4,x y z : τ_5,

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_5$$

関数 x より

$$\tau_2 = \tau_3 \to \tau_4 \to \tau_5$$

これらから

$$\tau_1 = (\tau_3 \to \tau_4 \to \tau_5) \to \tau_3 \to \tau_4 \to \tau_5$$

これの型は

```
('a -> 'b -> 'c) -> 'a -> 'b -> 'c
```

(ii) fun f x y z = x (y z) の型は
f: τ_1, x : τ_2, y: τ_3, z: τ_4, y z : τ_5, x (y z): τ_6,

$$\tau_1 = \tau_2 \to \tau_3 \to \tau_4 \to \tau_6$$

関数 x より

$$\tau_2 = \tau_5 \to \tau_6$$

関数 y より

$$\tau_3 = \tau_4 \to \tau_5$$

これらから

$$\tau_1 = (\tau_5 \to \tau_6) \to (\tau_4 \to \tau_5) \to \tau_4 \to \tau_6$$

すなわち

```
('a -> 'b) -> ('c -> 'a) -> 'c -> 'b
```

2. fun f x y z = x (y z) : int
これは (ii) 型で x (y z) つまり τ_6 が int. よって

```
('a -> int) -> ('b -> 'a) -> 'b -> int
```

3. fun f x y z = (x y z) : int
これは (i) 型で x y z つまり τ_5 が int. よって

```
('a -> 'b -> int) -> 'a -> 'b -> int
```

4. fun f x y z = x y (z: int)
   これは (i) 型で z つまりが τ_4 が int. よって

```
('a -> int -> 'b) -> 'a -> int -> 'b
```

5. fun f x y z = x (y z : int)
   これは (ii) 型で (y z) つまり τ_5 が int. よって

```
(int -> 'a) -> ('b -> int) -> 'b -> 'a
```

4 MLOO 問 3.11 の 1,2

1. fn x => x > 1
   x : τ_1, x > 1 : τ_2 とすると求めたいのは $\tau_1 \to \tau_2$. x は 1 と比較されているので τ_1 は int 型. また x > 1 は bool 型なので τ_2 は bool 型. よって
   int → bool.

2. fn x => fn y => fn z => (x y, x "Ada", y > z)
   x: τ_1, y: τ_2, z: τ_3, x y: τ_4, (x y, x "Ada", y > z) : τ_5 とすると求めたいのは $\tau_1 \to \tau_2 \to \tau_3 \to \tau_5$.
   まず τ_5 は組で τ_4 * τ_4 * bool. 関数 x の引数の型が同じということから τ_2 = string. また y と z で比較しているということから τ_3 = string. 関数 x について $\tau_1 = string \to \tau_4$.
   これらをすべてもとの式にいれると求める型は
   $(string \to \tau_4) \to string \to string \to \tau_4 * \tau_4 * bool$. よって

```
(string -> 'a) -> string -> string -> 'a * 'a * bool.
```

5 演習問題 5.6.4

```
fun comp F G =
  let
    fun C x =G(F(x))
  in
    C
  end;

fun add1 x = x + 1;
```

ここで関数 comp の型は

```
val comp = fn : ('a -> 'b) -> ('b -> 'c) -> 'a -> 'c
```

関数 add1 の型は

```
val add1 = fn : int -> int
```

a) val compA1 = comp add1;

関数. comp の第一引数が int → int なので, 'a と 'b が int. よって
(int→'a) → int → 'a
　注) この状態では処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

b) val compCompA1 = comp compA1;

compA1 が (int → 'a) → int → 'a
ということは、comp における
'a が int → 'a
'b も int → 'a
だから、答えは
((int → 'a) → 'b) → (int → 'a) → 'b
　注) この状態も処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

c) val f = compA1 add1;

compA1 が (int → 'a) → int → 'a add1 が int → int だから int → int
　内容としては f(x) は add1(x+1) つまり x + 2
　注)

```
- val compA1 : (int -> int) -> int -> int = comp add1;
- val f = compA1 add1;
```

とすれば処理系で計算できる.

5.1 d) f(2);

答えは 4.

e) val g = compCompA1 compA1;

compCompA1 の型: ((int→'a)→'b) → (int → 'a) → 'b
compA1 の型: (int→'a) → int → 'a
つまり compCompA1 の 'b は int → 'a.
よって関数 g の型は (int → 'a) → int → 'a
　注) この状態では処理系では型がわからないといって警告が出てそのままでは後の定義はできない.

f) val h = g add1;

型は int→int. この関数は引数に対し x+3 を行う.
　注)

```
- val compA1 : (int -> int) -> int -> int = comp add1;
- val compCompA1 : ((int -> int) -> int -> int) -> (int -> int) -> int -> int
 = comp compA1;
- val g : (int -> int) -> int -> int = compCompA1 compA1;
- val h = g add1;
```

とすれば処理系で計算できる.

g) h(2);

答えは 5.