

ソフトウェアサイエンス実験 S8 課題 2-5

200911434 青木大祐

平成 24 年 10 月 19 日

2.5.1 内部の関数が呼べないことを確かめる

以下にソースコードを示す。

```
1 type exp =
2   | IntLit of int
3   | Plus of exp * exp
4   | Times of exp * exp
5   | BoolLit of bool      (* 追加分; 真理値リテラル, つまり trueや false *)
6   | If of exp * exp * exp (* 追加分; if-then-else式 *)
7   | Eq of exp * exp      (* 追加分; e1 = e2 *)
8   | Greater of exp * exp ;;
9
10 (* 値の型 *)
11 type value =
12   | IntVal of int      (* 整数の値 *)
13   | BoolVal of bool    (* 真理値の値 *);;
14
15 let rec eval2b e =
16   let binop f e1 e2 =
17     match (eval2b e1, eval2b e2) with
18     | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
19     | _ -> failwith "integer values expected"
20   in
21   match e with
22   | IntLit(n) -> IntVal(n)
23   | Plus(e1,e2) -> binop (+) e1 e2
24   | Times(e1,e2) -> binop ( *) e1 e2
25   | _ -> failwith "unknown expression";;
26
27 binop (+) (IntLit 10) (IntLit 20);;
```

これを実行すると、以下のようなエラーが出力され、binop 関数が呼び出せないことが分かる。

```
1 # type value = IntVal of int | BoolVal of bool
2 # val eval2b : exp -> value = <fun>
3 # Characters 1-6:
4 binop (+) (IntLit 10) (IntLit 20);;
5 ^^^^^
6 Error: Unbound value binop
7 #
```

2.5.2 外から使えない理由

内部で定義した関数が外から呼び出せない利点として、名前を汚染しないことが考えられる。例えばこのプログラムを外部から参照して利用するとして、偶然 binop という関数を定義してしまうと、名前が衝突してしまいコンパイルできず、これを解消するために別の(最適でない)名前を考える必要が生じる。しかし、内部で定義した関数の名前が外にもれないことで、この問題は解決することが出来る。

2.5.3 binop の型

binop を外に出した形のソースコードは以下のとおり。

```
1 (* eval2b : exp -> value *)
2
3 type exp =
4   | IntLit of int
5   | Plus of exp * exp
6   | Times of exp * exp
7   | BoolLit of bool      (* 追加分; 真理値リテラル, つまり trueや false *)
8   | If of exp * exp * exp (* 追加分; if-then-else式 *)
9   | Eq of exp * exp      (* 追加分; e1 = e2 *)
10  | Greater of exp * exp ;;
11
12 (* 値の型 *)
13 type value =
14   | IntVal of int      (* 整数の値 *)
15   | BoolVal of bool    (* 真理値の値 *);;
16
17 let rec eval2b e =
18   let binop f e1 e2 =
19     match (eval2b e1, eval2b e2) with
20     | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
21     | _ -> failwith "integer values expected"
22   in
23   match e with
24   | IntLit(n) -> IntVal(n)
```

```

25 | Plus(e1,e2) -> binop (+) e1 e2
26 | Times(e1,e2) -> binop ( * ) e1 e2
27 | _ -> failwith "unknown expression";;
28
29 eval2b (Plus ((IntLit 10), (IntLit 20)));;
30
31 let binop f e1 e2 =
32   match (eval2b e1, eval2b e2) with
33   | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
34   | _ -> failwith "integer values expected";;
35
36 binop (+) (IntLit 10) (IntLit 20);;

```

これを実行すると、以下の様な結果が得られる。

```

1 #           type value = IntVal of int | BoolVal of bool
2 #               val eval2b : exp -> value = <fun>
3 #   - : value = IntVal 30
4 #           val binop : (int -> int -> int) -> exp -> exp -> value = <fun>
5 #   - : value = IntVal 30
6 #

```

これより、binop の型は $(int \rightarrow int \rightarrow int) \rightarrow exp \rightarrow exp \rightarrow value = \langle fun \rangle$ であることが分かる。