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。

```
type exp =
  | IntLit of int
  | Plus of exp * exp
  | Times of exp * exp
  | BoolLit of bool        (* 追加分; 真理値リテラル, つまり trueや false  *)
  | If of exp * exp * exp  (* 追加分; if-then-else式 *)
  | Eq of exp * exp        (* 追加分; e1 = e2 *)
  | Greater of exp * exp ;;

(* 値の型 *)
type value =
  | IntVal  of int          (* 整数の値 *)
  | BoolVal of bool         (* 真理値の値 *);;

let rec eval2b e =
  let binop f e1 e2 =
    match (eval2b e1, eval2b e2) with
      | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
      | _ -> failwith "integer values expected"
  in
  match e with
    | IntLit(n)    -> IntVal(n)
    | Plus(e1,e2)  -> binoop (+) e1 e2
    | Times(e1,e2) -> binop ( * ) e1 e2
    | _ -> failwith "unknown expression";;

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。

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

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。

```
(* eval2b : exp -> value *)

type exp =
  | IntLit of int
  | Plus of exp * exp
  | Times of exp * exp
  | BoolLit of bool        (* 追加分; 真理値リテラル, つまり trueや false  *)
  | If of exp * exp * exp  (* 追加分; if-then-else式 *)
  | Eq of exp * exp        (* 追加分; e1 = e2 *)
  | Greater of exp * exp ;;

(* 値の型 *)
type value =
  | IntVal  of int          (* 整数の値 *)
  | BoolVal of bool         (* 真理値の値 *);;

let rec eval2b e =
  let binop f e1 e2 =
    match (eval2b e1, eval2b e2) with
      | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
      | _ -> failwith "integer values expected"
  in
  match e with
    | IntLit(n)    -> IntVal(n)
```

```
    | Plus(e1,e2)  -> binop (+) e1 e2
    | Times(e1,e2) -> binop ( * ) e1 e2
    | _ -> failwith "unknown expression";;

eval2b (Plus ((IntLit 10), (IntLit 20)));;

let binop f e1 e2 =
  match (eval2b e1, eval2b e2) with
    | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
    | _ -> failwith "integer values expected";;

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。

```
#         type value = IntVal of int | BoolVal of bool
#                       val eval2b : exp -> value = <fun>
#   - : value = IntVal 30
#         val binop : (int -> int -> int) -> exp -> exp -> value = <fun>
#   - : value = IntVal 30
#
```

これより、binop の型は $(int- > int- > int)- > exp- > exp- > value =< fun >$ であることが分かる。