

ソフトウェアサイエンス実験 S8 課題 3-2

200911434 青木大祐

平成 24 年 11 月 8 日

eval3 のソースコードを以下に示す。また、作成したソースコード全体は末尾に添付する。

```
1 let rec eval3 ?(mode=0) e env =                (* env を引数に追加 *)
2   if mode != 0 then print_env env;              (* デバッグモード *)
3
4   let binop f e1 e2 env =                      (* binop の中でも eval3 を呼ぶので env を追加 *)
5     match (eval3 e1 env ~mode, eval3 e2 env ~mode) with
6     | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
7     | _ -> failwith "integer value expected"
8   in
9   match e with
10  | Var(x)      -> lookup x env
11  | IntLit(n)   -> IntVal(n)
12  | BoolLit(b)  -> BoolVal(b)
13  | Plus(e1,e2) -> binop (+) e1 e2 env          (* env を追加 *)
14  | Times(e1,e2) -> binop ( * ) e1 e2 env        (* env を追加 *)
15  | Eq(e1,e2) ->
16    begin
17      match (eval3 e1 env ~mode, eval3 e2 env ~mode) with
18      | (IntVal(n1),IntVal(n2)) -> BoolVal(n1=n2)
19      | (BoolVal(b1),BoolVal(b2)) -> BoolVal(b1=b2)
20      | _ -> failwith "wrong value"
21    end
22  | If(e1,e2,e3) ->
23    begin
24      match (eval3 e1 env ~mode) with                (* env を追加 *)
25      | BoolVal(true) -> eval3 e2 env                (* env を追加 *)
26      | BoolVal(false) -> eval3 e3 env                (* env を追加 *)
27      | _ -> failwith "wrong value"
28    end
29  | Let(x,e1,e2) ->
30    let env1 = ext env x (eval3 e1 env ~mode)
31    in eval3 e2 env1 ~mode
32  | _ -> failwith "unknown expression";;
```

3.1.1 動作の確認

次のような例を与えて動作を確認した。

```
1 eval3 (Let ("x", IntLit 1, (Plus (IntLit 2, Var "x")))) [];;
2 eval3 (Let ("x", IntLit 1, Let("y", IntLit 3, (Plus (Var "y", Var "x"))))) [];;
3 eval3 (Let ("x", BoolLit true, If(Eq(Var "x", BoolLit true), IntLit 1, IntLit 2))) [];;
```

実行結果は以下の通り。

```
1 # - : value = IntVal 3
2 # - : value = IntVal 4
3 # - : value = IntVal 1
```

正しく実行できていることが分かる。

3.1.2 let のネスト

次のような式で、let 文をネストした際の動作を確認した。

```
1 eval3 (Let ("x", IntLit 1, (Let ("x", IntLit 2, Var "x")))) [];;
```

実行結果は以下のとおり。

```
1 # - : value = IntVal 2
```

また、Ocaml で実行した結果は以下のとおり。同じように計算できていることが分かる。

```
1 # let x = 1 in let x = 2 in x;;
2 Warning 26: unused variable x.
3 - : int = 2
```

3.1.3 let のスコープ

次のような例を与えて動作を確認した。

```
1 eval3 ~mode:1 (Let ("x", IntLit 1, (Let ("y", Plus(Var "x", IntLit 1), Plus(Var "x", Var "y"))))) [];;
```

実行結果は以下のとおり。

```
1 [][] [x = 1;] [x = 1;] [x = 1;] [x = 1;] [y = 2; x = 1;] [y = 2; x = 1;] [y = 2; x = 1;] - : value = IntVal 3
```

eval の中で次の eval が呼ばれる時に現在の env を渡しているので、内側の式でも外側で定義した変数が参照できていることが分かる。

3.1.4 束縛の解除

Let 文では、受け取った環境について新たな束縛を追加した環境を新しく作り、内側の式には新しい環境を適用している。そのため、Let 文の外側の環境には影響を及ぼさない仕組みになっている。

3.1.5 ソースコード全体

```
1  (* eval2b : exp -> value *)
2  let debug = false;
3
4  type exp =
5  | IntLit of int
6  | Plus of exp * exp
7  | Times of exp * exp
8  | BoolLit of bool      (* 追加分; 真理値リテラル, つまり trueや false *)
9  | If of exp * exp * exp (* 追加分; if-then-else式 *)
10 | Eq of exp * exp      (* 追加分; e1 = e2 *)
11 | Greater of exp * exp
12 | Var of string
13 | Let of string * exp * exp ;;
14
15 (* 値の型 *)
16 type value =
17 | IntVal of int      (* 整数の値 *)
18 | BoolVal of bool    (* 真理値の値 *);;
19
20 let emptyenv () = [];;
21
22 let ext env x v = (x,v) :: env;;
23
24 let rec lookup x env =
25   match env with
26   | [] -> failwith ("unbound variable: " ^ x)
27   | (y,v)::tl -> if x=y then v
28                   else lookup x tl;;
29
30 let string_of_env env =
31   let string_of_val (e:value) : string =
32     match e with
33     | IntVal n -> string_of_int n
34     | BoolVal true -> "true"
35     | BoolVal false -> "false"
36   in
37   let rec internal (env:(string * value) list) =
38     match env with
39     | [] -> ""
40     | (name, v)::rest -> name ^ " = " ^ (string_of_val v) ^ ";" ^ (internal rest)
41   in
42   "[" ^ (internal env) ^ "]";;
43
44 (* eval3 : exp -> (string * value) list -> value *)
45 (* let と変数、環境の導入 *)
46 let print_env (env: (string * value) list) =
47   print_string( string_of_env env);;
48
49 let rec eval3 ?(mode=0) e env =          (* env を引数に追加 *)
50   if mode != 0 then print_env env;
51
52   let binop f e1 e2 env =              (* binop の中でも eval3 を呼ぶので env を追加 *)
53     match (eval3 e1 env ~mode, eval3 e2 env ~mode) with
54     | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
55     | _ -> failwith "integer value expected"
56   in
57   match e with
58   | Var(x) -> lookup x env
59   | IntLit(n) -> IntVal(n)
60   | BoolLit(b) -> BoolVal(b)
61   | Plus(e1,e2) -> binop (+) e1 e2 env    (* env を追加 *)
62   | Times(e1,e2) -> binop ( * ) e1 e2 env  (* env を追加 *)
63   | Eq(e1,e2) ->
64     begin
65       match (eval3 e1 env ~mode, eval3 e2 env ~mode) with
66       | (IntVal(n1),IntVal(n2)) -> BoolVal(n1=n2)
67       | (BoolVal(b1),BoolVal(b2)) -> BoolVal(b1=b2)
68       | _ -> failwith "wrong value"
69     end
70   | If(e1,e2,e3) ->
71     begin
72       match (eval3 e1 env ~mode) with          (* env を追加 *)
73       | BoolVal(true) -> eval3 e2 env    (* env を追加 *)
74       | BoolVal(false) -> eval3 e3 env   (* env を追加 *)
75       | _ -> failwith "wrong value"
76     end
77   | Let(x,e1,e2) ->
78     let env1 = ext env x (eval3 e1 env ~mode)
79     in eval3 e2 env1 ~mode
80   | _ -> failwith "unknown expression";;
81
82
83 (* 3.2.1 *)
84 eval3 (Let ("x", IntLit 1, (Plus (IntLit 2, Var "x")))) [];;
85 eval3 (Let ("x", IntLit 1, Let("y", IntLit 3, (Plus (Var "y", Var "x"))))) [];;
86 eval3 (Let ("x", BoolLit true, If(Eq(Var "x", BoolLit true), IntLit 1, IntLit 2))) [];;
87 (* 3.2.2 *)
```

```
88 eval3 (Let ("x", IntLit 1, (Let ("x", IntLit 2, Var "x")))) [];;
89 (* 3.2.3 *)
90 eval3 ~mode:1 (Let ("x", IntLit 1, (Let ("y", Plus(Var "x", IntLit 1), Plus(Var "x", Var "y"))))) [];;
91 (* 3.2.4 *)
92 eval3 (Let ("x", IntLit 1, Plus(Let("x", IntLit 2, Plus(Var "x", IntLit 1)), Times(Var "x", IntLit 2)))) [];;
93
94 print_env [("x", IntVal 1); ("y", IntVal 2); ("z", BoolVal true)]
```