ソフトウェアサイエンス実験 S8 課題 5-2

200911434 青木大祐

平成 24 年 11 月 9 日

5.2.1 eval4 の実装

```
let x = 1 in
  let f = fun y -> x + y in
    let x = 2 in
      f (x + 3)
```

上記のプログラムを、ミニ Ocaml のプログラムとして実行した。結果を以下に示す。

```
- : Syntax.value = Syntax.IntVal 6
```

5.2.2 APP の変更

上記のプログラムについて、*eval*4 内の *env*1 を *env* に変更した際の実行結果を示す。

```
- : Syntax.value = Syntax.IntVal 7
```

env はクロージャに保持されている環境ではなく、関数適用が行われた時点での環境なので、これは動的束縛だといえる。

5.2.3 factorial の定義

ある数の階乗を求めるには、ループもしくは再帰の構造が必要になる。ミニ OCaml ではループ構文がまだ用意されていないので、再帰を使うことを考える。
しかしミニ OCaml では、関数抽象を行う時点での環境にその関数自体は含まれていないため、”*unboundvariable* : *fact*” のように未定義の変数であるというふうにエラーが出てしまう。
よって、ここまでの段階では factorial を定義することは不可能だと考えられる。

5.2.4 評価順序

(*e*1 + *e*2) について、実際に評価される部分は以下の二行目である。

```
  let binop f e1 e2 env =       (* binop の中でも eval4 を呼ぶので env を追加 *)
    match (eval4 e1 env ~mode, eval4 e2 env ~mode) with
    | (IntVal(n1),IntVal(n2)) -> IntVal(f n1 n2)
    | _ -> failwith "integer value expected"
  in
```

また、Ocaml で以下のプログラムを実行すると次のような結果が得られる。

```
#   (print_string "1"; 10) + (print_string "2"; 20);;
21- : int = 30

# (print_int 1, print_int 2);;
21- : unit * unit = ((), ())
```

この結果から、カンマ区切りの式は後者が先に評価されるので、(*e*1 + *e*2) については OCaml とミニ OCaml で違いは無いと考えられる。

(*e*1*e*2) という形の関数適用について、例として提示されたプログラムを実行すると、以下の様な結果が得られた。

```
#   (print_string "1"; (fun x -> x)) (print_string "2"; 20) ;;
21- : int = 20
```

これは、適用する関数本体より先に引数を評価していることを表している。しかし、現時点でのミニ OCaml の実装は以下のとおりである。

```
  | App(e1,e2) ->
     begin
      match (eval4 e1 env) with
        | FunVal(x,body,env1) ->
            let arg = (eval4 e2 env)
      in eval4 body (ext env1 x arg)
        | _ -> failwith "function value expected"
     end
```

これを次のように変更することで、OCaml と同じ評価順序になる eval4 関数が得られる。

```
  | App(e1,e2) ->
     begin
      let arg = (eval4 e2 env) in
      match (eval4 e1 env) with
        | FunVal(x,body,env1) ->
      eval4 body (ext env1 x arg)
        | _ -> failwith "function value expected"
     end
```