ソフトウェアサイエンス実験 S8 課題 2-3

200911434 青木大祐

平成 24 年 9 月 14 日

2.3.1 eval

eval1 関数を用いて以下のように式を評価した。

```
type exp =
  | IntLit of int          (* リテラル *)
  | Plus of exp * exp      (* 足し算 *)
  | Times of exp * exp     (* かけ算 *)
  | Sub of exp * exp       (* 引き算 *)
  | Div of exp * exp       (* 割り算 *);;

let rec eval1 e =
  match e with
  | IntLit(n)    -> n
  | Plus(e1,e2)  -> (eval1 e1) + (eval1 e2)
  | Times(e1,e2) -> (eval1 e1) * (eval1 e2)
  | _ -> failwith "unknown expression";;

(* 以下の式に対してテストを行った *)
let easy:exp = Plus(Times(IntLit 2, IntLit 2), Plus(IntLit 3, IntLit (-4)));;
let sample:exp =  Div(Plus(IntLit 1, Times(IntLit (-2), IntLit 5)), Sub(IntLit 4, IntLit (-3)));;

eval1 easy;;
eval1 sample;;
```

リスト 1: 実行結果

```
   # val sample : exp =
     Div (Plus (IntLit 1, Times (IntLit (-2), IntLit 5)),
     Sub (IntLit 4, IntLit (-3)))
   #   - : int = 3
   # Exception: Failure "unknown expression".
   #
```

正しく解釈できており、また解釈できない場合においては例外が発生している。

2.3.2 match の順序

以下の様なソースコードを用いて、match の順序について検証した。

```
type exp =
  | IntLit of int          (* リテラル *)
  | Plus of exp * exp      (* 足し算 *)
  | Times of exp * exp     (* かけ算 *)
  | Sub of exp * exp       (* 引き算 *)
  | Div of exp * exp       (* 割り算 *);;

let rec eval1 e =
  match e with
  | IntLit(n)    -> n
  | Plus(e1,e2)  -> (eval1 e1) + (eval1 e2)
  | _ -> failwith "unknown expression"
  | Times(e1,e2) -> (eval1 e1) * (eval1 e2);;

let test:exp = Times(IntLit 1, IntLit 1);;
eval1 test;;
```

リスト 2: 実行結果

```
 # Exception: Failure "unknown expression".
```

Times は末尾のパターンにマッチして欲しいところだが、先に_にマッチするので Times の解釈で例外が発生している。

2.3.3 引き算と割り算の追加

以下のように定義を修正し、引き算と割り算についても計算できるようにした。また、ゼロ除算については解釈できなかった場合と同様に例外を発生させるようにした。

```
type exp =
  | IntLit of int          (* リテラル *)
  | Plus of exp * exp      (* 足し算 *)
  | Times of exp * exp     (* かけ算 *)
  | Sub of exp * exp       (* 引き算 *)
  | Div of exp * exp       (* 割り算 *);;

let rec eval1 e =
  match e with
  | IntLit(n)    -> n
  | Plus(e1,e2)  -> (eval1 e1) + (eval1 e2)
  | Times(e1,e2) -> (eval1 e1) * (eval1 e2)
  | Sub(e1, e2)  -> (eval1 e1) - (eval1 e2)
  | Div(e1, IntLit 0) -> failwith "divide by zero"
  | Div(e1, e2) -> (eval1 e1) / (eval1 e2)
  | _ -> failwith "unknown expression";;

let easy:exp = Plus(Times(IntLit 2, IntLit 2), Plus(IntLit 3, IntLit (-4)));;
let sample:exp =  Div(Plus(IntLit 1, Times(IntLit (-2), IntLit 5)), Sub(IntLit 4, IntLit (-3)));;

eval1 easy;;
eval1 sample;;
```

リスト 3: 実行結果

```
#   - : int = 3
# - : int = -1
# Exception: Failure "divide by zero".
```

正しく計算できている事がわかる。また、ゼロ除算についても例外が発生している。