主専攻実験[S-8] 関数プログラミング

課題2-4, 2-5

情報科学類 202113564 三村潤之介

# 課題 2-4.

・テストプログラムをすべて実行しなさい。

示されているプログラムリストはすべてエラー無く実行された。

例えば、

let \_ = eval2 (If (Eq(IntLit 2, IntLit 11),

Times(IntLit 1, IntLit 2),

Times(IntLit 1, Plus(IntLit 2,IntLit 3))))

を見てみる。

実行結果は以下のようになる。

# let \_ = eval2(If (Eq(IntLit 2, IntLit 11), Times(IntLit 1, IntLit 2), Times(IntLit 1, Plus(IntLit 2, IntLit 3))));;

- : value = IntVal 5

eval2によって、Ifは3つの引数をとり、1つめの引数が真ならば2つめの引数、偽ならば3つめの引数を返すよう定義されている。この例では、1つめの引数として、Eq文が評価され、これが偽であるため、3つめの引数である(1 \* (2 + 3))を示すTimes文が評価されたものが返されている。

・エラーが起こる例について、調べなさい。

エラー起こるというのは、ここでは、failwith文が与えられているケースにmatchすることである。

一番外側のmatch文の”unknown expression”について、これにmatchする式を考える。「Plus(IntLit 1, IntLit 2**, IntLit 3**)」や「**Prus**(IntLit 1, IntLit2)」などが該当するだろうか、と予想したが、これらはそもそもexpで定義されていないため、eがexpであると予期されているeval2 eでは「eがexpでないため」という理由でエラーになってしまう。そのため、外側のmatchでパターンマッチングが発生しながら、例外にmatchする式は無いと考える。このことは、コンパイル時に

53 | | \_ -> failwith "unknown expression"

^

Warning 11: this match case is unused.

のように警告が出ることからも同様に考えられる。

その他、それぞれの評価部内の”wrong value”例外については、命令や仮引数の数が対応していながら、その仮引数の内部が予期される値の型と整合しない場合に発生すると考える。

・整数値と真理値をEqで比較する式をeval2に与えると、何が起きるか。

実行結果は以下のようになる。

# eval2(Eq(IntLit 1, BoolLit true));;

Exception: Failure "wrong value".

前述したとおり、Eq文では中の仮引数がどちらも整数値またはどちらも真理値であることが予期されるため、型が整合せずにEq文内の”wrong value”例外にパターンマッチする。

・整数同士を比較する式としてGreater(e1, e2)を追加したときの処理を追加しなさい。

Greaterを実装するためには、まずGreater文がexpとして扱われる必要がある。

仮引数はどちらもexpであるため、expの定義部に以下のように追加する。

type **exp** **=**

**|** **IntLit** of int

…

**|** **Greater** of exp **\*** exp

次にeval2内において、実際の処理を記述する。Greater(e1, e2)において、e1, e2は整数の値であるので、どちらも整数であるときに評価された仮引数の値を比較してBool 値を返し、片方または両方が真理値であるときは”wrong value”として例外を返せば良い。

以下のように記述した。

**|** **Greater**(*e1***,** *e2*) **->**

begin

match (eval2 e1, eval2 e2) with

**|** (**IntVal**(*n1*)**,** **IntVal**(*n2*)) **->** **BoolVal**(n1>n2)

**|** \_ **->** failwith "wrong value"

end

**|** \_ **->** failwith "unknown expression"

例を与えて実行した結果を示す。

# eval2(Greater(IntLit 3, IntLit 4));;

- : value = BoolVal false

# eval2(Greater(IntLit 4, IntLit 2));;

- : value = BoolVal true

上部は3>4の値であるFalse、下部は4>2の値であるTrueがBool値で返されていることがわかる。

# 課題2-5

・binopという関数は、eval2bの外からは使えないことを確かめよ。

binopが外部から使えるならば、コンソールモード(?)で例えば以下のような式

binop((+) IntVal 3, IntVal 2);;

によって、IntVal(5)が返ってくるはずである。しかし、実際には、

# binop((+) IntVal 3, IntVal 2);;

Error: Unbound value binop

となり、binopが開放されている変数としてエラーが出てしまう。これはeval2bの外ではbinopが関数として定義されていないことを示す。

・「外から使える」ことがメリットではなく、「外から使えないこと」がメリットとして主張されている理由を考えよ。

実験資料2-5.より、以下の記述を引用する。

このようにすることにより、eval2b の外から binop という関数は使えない。 従って、他に binop という名前の関数があっても名前が衝突しない、というメリットがある。

おそらくこれがその大きなメリットであると考える。

「外から使えないこと」自体のメリットではないが、このように記述することのメリットとしては、カプセル化との類似性があることから類推して、「使用者が内部構造を理解しなくとも利用可能である」こともメリットに挙げられるだろう。この例で言えば、使用者はbinopの厳密な定義を理解しなくとも、二項演算子及びオペランドを与えることさえ知っていれば十分に利用できる。

・eval2 に対して、eval2b への修正をほどこし、 eval2b を完成させよ。

eval2bの大きな特徴としてはbinopによって二項演算子を渡すだけで二項演算ができるようになっていることである。Greater, Eq, Ifのうち、二項演算であるのはGreaterおよびEqであるため、それらについて同様な処理を行うことを考えた。

最終的に、Greaterに対してbinop\_boolという関数を定義することにした。binop f e1 e2自体は(int -> int -> int) -> exp -> exp -> valueであるため、二項演算子にあたるfはint -> int -> intとして型が推定される。よって、int -> int -> boolを望む「>」と同じ関数で扱うことができないと考えたためである。

binop\_bool f e1 e2は(int -> int -> bool) -> exp -> exp -> valueであるよう定義し、「>」や「<」などのboolを出力とする二項演算子はこれを用いて評価するようにした。これによってGreaterの他、(a < b)を評価するLess(a, b)も実装できるようになった。

Eqについては、「=」が引数によって(int -> int -> bool)と(bool -> bool -> bool)のどちらであるかが変わってくるため、いわゆるカプセル化が難しいと考え、eval2の実装をそのまま流用した。

Ifについては、前述したとおり二項演算子の類いではないため、eval2の実装をそのまま流用した。

・binop の定義を eval2b の中から外に出し、その型がどうなるかを調べよ。

binopをそのままeval2b外側にコピーした。以下のように出力された。

# binop;;

- : (int -> int -> int) -> exp -> exp -> value = <fun>

これは、intを返す二項演算子、及び２つのexpからvalueを返す式であることを示している。ここで、binopに(+)や(>)などの演算子を渡してみる。

# binop (+);;

- : exp -> exp -> value = <fun>

# binop (>);;

Error: This expression has type int -> int -> bool

but an expression was expected of type int -> int -> int

Type bool is not compatible with type int

上部は、先程の(int -> int -> int)部をとった形に等しくなっている。一方で、下部はエラーが出力されている。これは(int -> int -> bool)である「>」が整合していないことを示している。