主専攻実験[S-8] 関数プログラミング

課題7-4

情報科学類 202113564 三村潤之介

# 課題7-4

・関数tcheck3 に、いくつかの式を与え、正しく型検査できていることを確かめよ。

資料にある5つの例を与えた。

* 式(fun x -> if true then x else 100) と型環境 [("x",TInt)]
* 式(fun x -> if true then x else 100) と型環境 [("x",TBool)]
* 式((fun x -> if true then x else 100) (if true then y else 200)) と型環境[("x",TInt);("y",TInt)]
* 式(fun f -> (fun x -> f (f (f x + 10)))) と適当な型環境
* 式(fun f -> (fun g -> (fun x -> f (g x)))) と適当な型環境

以下はその結果である。

# (tcheck3 [("x",TInt)] (Fun("x", (If(BoolLit(true), Var("x"), IntLit(100))))));;

- : ty = TArrow (TInt, TInt)

intを受け取ってintを返す(int->int)を返す。

# (tcheck3 [("x",TBool)] (Fun("x", (If(BoolLit(true), Var("x"), IntLit(100))))));;

Exception: Failure "type error in IF".

xがintでないとIf文は成立しないが、型環境内ではxがboolとなっているためエラーとなる。

# (tcheck3 [("x",TInt);("y",TInt)] (App(Fun("x", If(BoolLit(true), Var("x"), IntLit(100))), (If(BoolLit(true), Var("y"), IntLit(200))))));;

- : ty = TInt

外側のIf内の第二引数、第三引数がどちらもintであるため、intを返す。

# (tcheck3 [("x",TInt); ("f",TArrow(TInt, TInt))] (Fun("f", Fun("x", App(Var("f"), App(Var("f"), App(Var("f"), Plus(Var("x"), IntLit(10)))))))));;

- : ty = TArrow (TArrow (TInt, TInt), TArrow (TInt, TInt))

fを(int->int),xをintと型付けすると、(int->int)->(int->int)が返される。

# (tcheck3 [("f",TArrow(TInt, TInt));("g",TArrow(TInt, TInt));("x",TInt)] (Fun("f", Fun("g", Fun("x", App(Var("f"), App(Var("g"), Var("x"))))))));;

- : ty =

TArrow (TArrow (TInt, TInt),

TArrow (TArrow (TInt, TInt), TArrow (TInt, TInt)))

f=(tf1->tf2),g=(tg1->tg2),x=txとすると、tx=tg1 && tg2=tf1であれば型が整合する。この例では、(int->int)->((int->int)->(int->int))が返される。

・e2とe3 の型が等しい場合でも型検査に通るよう、tcheck3 における if文の処理を拡張せよ。

第一引数がTBoolで、 第二引数と第三引数が等しければ良いので以下のように書き換える。

| If(e1,e2,e3) ->

begin

match (tcheck3 te e1, tcheck3 te e2, tcheck3 te e3) with

| (TBool, t2, t3) ->

if t2 = t3 then t2

else failwith "type error in IF"

| \_ -> failwith "type error in IF"

end

以下は実行例である。

# (tcheck3 [("x",TInt);("y",TInt)] (If(BoolLit(true), Fun("x", Plus(Var("x"), IntLit(1))), Fun("y", Plus(Var("y"), Var("y"))))));;

- : ty = TArrow (TInt, TInt)

Ifの第二引数と第三引数を(int->int)とし、(int->int)が返ってきていることがわかる。

・[必須でない課題] tcheck3を拡張して、let式に対応させよ。

eval式と同様に、環境を拡張して再検査させることにした。そのために、環境を拡張するextをそのまま流用できる。

let ext env x v = (x,v) :: env

これを用いて、以下のように記述できる。

| Let(x, e1, e2) ->

let ne = (ext (te) (x) (tcheck3 te e1)) in

tcheck3 ne e2

以下は実行例である。

# (tcheck3 [("x",TInt); ("y",TInt)] (Fun("x", Let("y", Plus(Var("x"), IntLit(10)), Plus(Var("y"), IntLit(5))))));;

- : ty = TArrow (TInt, TInt)

以上のように実行できた。

・[必須でない課題]型検査のまとめとして、tcheck3 を拡張して、ミニOCaml言語のなるべく多くの構文に対応するようにせよ。

evalにあったもので、Times,Greaterを実装した。実装それぞれ、TimesはAdd, GreaterはEqと似ているため、実装の記載を省略し、実行例を示す。

# (tcheck3 [("x",TInt); ("y",TInt)] (Times(Var("x"),Var("y"))));;

- : ty = TInt

# (tcheck3 [("x",TBool); ("y",TBool)] (Greater(Var("x"),Var("y"))));;

- : ty = TBool

以上のようにTimes, Greaterを実装できた。