計算機科学実験演習３　ソフトウェア　レポート３

氏名：吉村仁志

提出日：７月１９日（木）

第1章　ML^2インタプリタ型推論

1.1　Exercise　4.2.1

chap04-1.pdf で解説されていない部分について解説する。まず、ty_prim についてだが、Plus の時と同様に、受け取った２つの ty に関して op に対して望まれる型であるときに、その op の結果として来るべきである型を返せば良く、そうでない時にはエラーを発生させれば良い。Mult は TyInt と TyInt を受け取って TyInt を返し、Lt は TyInt と TyInt を受け取って TyBool を返す。といった具合である。AAND、OOR は引数も結果もすべて TyBool である。次に、ty_exp について説明する。IfExp についてだが、型の条件としては、exp1 の型が TyBool であり、exp2 と exp1 の型が等しいということである。そこで、まず tyarg1 に ty_exp を再帰呼出しして exp1 の型を束縛し、match 文でそれが TyBool 出ない時はエラーを発生させる。TyBool であるもとで、exp2 と exp1 の型を評価し、それが等しいかどうかを match 文で評価しそうでない時はエラーを発生させる。等しいというもとで、最終的には exp2 もしくは exp3 の型が結果になるので、それを返すようにする。LetInExp は、まず id を exp1 の型の評価結果で束縛し、それで現在の環境を拡張した新たな環境 tynewenv を作成し、そのもとで exp2 の型を評価すれば良い。

第2章　ML^3インタプリタ型推論

2.1　Exercise　4.3.1

まず pp_ty の実装についてだが、これは string_of_ty を呼び出すだけと定義したので、string_of_ty について解説する。これは、ty 型の値を受け取ってそれを文字列にするような関数である。match 構文を用いて型によって場合分けする。まず TyInt、TyBool のときはそれぞれ'int'、'bool' をそのまま返せば良い。TyFun(ty1, ty2) のときは、ty1、ty2 をそれぞれ再び評価しなければならなく、'TyFun(' (string_of_ty ty1) (string_of_ty ty) ')'といったように中身について string_of_ty を再帰呼出しをする。TyList ty のときは、ty を再び評価するので、(string_of_ty ty) 'list' というようにする。TyVar num のときは、その num の中身を探しに行くことになる。そこで、すでに定義された TyVar num を格納しておくものが必要であり、syntax.ml にグローバル変数としてリストの参照 tyvarlist * tyvar * id を定義しておくこれは、tyvar とそれを表す文字を対応させている。そこで、TyVar num のときは、tyvarlist を走査して num に対応する文字を返すようにする。また、リストを走査するための関数 researchlist を定義してそれを利用した。次に fresh_tyvar の実装だが、まず型変数が現れうるのは、TyVar num、TyFun、TyList である。そこで、match 構文を利用してそれぞれの型に関して処理を行う。TyVar num のときは num は欲しいものそのものであるから、MySet.ml で定義されている関数 insert を利用して、insert num MySet.empty とした MySet 型の集合を作り、それを返す。TyFun(ty1, ty2) のときは、ty1、ty2 のそれぞれを評価する必要があるので、ty1、ty2 を再帰的に評価しその結果を統合すれば良い。よって、MySet.ml で定義されている関数 union を利用して、union (fresh_tyvar ty1) (fresh_tyvar ty2) を返せば良い。TyList ty のときは ty を評価する必要があり、fresh_tyvar ty を返す。それ以外の型のときは MySet.empty を返す。

2.2　Exercise　4.3.2

subst_type は、(tyvar * ty) list と ty を受け取って (tyvar * ty) list を ty に適用するのだが、ここでそのための補助関数として、sub_subst を定義する。これは、(tyvar * ty) と ty を受け取って (tyvar * ty) を ty に適用しその結果を返すような関数である。これは match 構文を利用して、それぞれの型

に関して処理を行う。TyInt、TyBool ときはそのまま TyInt、TyBool を返せば良い。TyFun(ty1, ty2) 、TyList ty のときは前者は ty1、ty2 に関して、後者は ty に関して再帰呼出しする必要があるので、TyFun(sub_subst ... ty1, sub_subst ... ty2) 、TyList sub_subst ... ty をそれぞれ返せば良い。TyVar num のときは、num が引数として受け取った tyvar と一致するときには型を置き換える必要がある。sub_subst の引数を (numm, tyy) ty とすると、num = numm のときは、tyy を返し、そうでないときはそのまま Tyvar num を返すといったようにする。これで sub_subst の実装は終了する。subst_type であるが、これは受け取った第一引数である、(tyvar * ty) list の一要素毎に sub_subst を呼びだせば良い。これで subst_type の実装は終了する。

2.3 Exercise 4.3.3

まず、型 ty の中に TyVar num が現れるかどうかを判定するための関数 reserarch_ftv を定義する。これは num と ty を受け取って、bool 型の値を返すような関数である。存在するとき true を返し、そうでないとき false を返す。受け取る引数を、numm ty とし、match 構文で ty の型によってそれぞれ処理を行う。TyVar num のときは、numm = num を返せば良い。TvFun(ty1, ty2) のときは、ty1、ty2 のそれぞれを評価する必要があり、ty1、ty2 のどちらかが true のときには全体の結果も true になるので、(research_ftv ty1) || (research_ftv ty2) を返せば良い。TyList ty のときは、ty を評価すれば良く、research_ftv ty を返せば良い。それ以外のときは false を返す。これで research_ftv の実装は終了する。次に、型代入のリスト numtyl と、ty のタプルのリスト tytyl を受け取って、後者に前者を適用するような関数 map_subst を定義する。これは、tytyl が (ty1, ty2) :: rest という形になるとき、ty1、ty2 のそれぞれに型代入を適用すればよく、このとき ((subst_type numtyl ty1), (subst_type numtyl ty2)) :: (map_subst rest) を返すようにする。tytyl が空リストのときは空リストを返す。次に unify について解説する。受け取る引数は tylist であるとする。解説にあるとおり、tylist の型によって場合分けをする。tylist が空リストのときは空リストを返せば良い。そうでない時、すなわち (t1 , t2) :: rest にマッチしたとする。t1 = t2 のときは条件は満たされており、unify rest を返せば良い。以下そうでない時について考える。t1、t2 がそれぞれ TyFun(tyy11, tyy12)、TyFun(tyy21, tyy22) のとき、条件は tyy11 = tyy21、tyy12 = tyy 22 であるので、解説にあるとおり、unify ((tyy11, tyy21) :: (tyy12, tyy22) :: rest) を返す。t1、t2 がそれぞれ TyVar num 、tyy のとき、research_ftv num tyy を評価して、解説にあるとおり、これが true のときはエラーを発生させる。false のとき、解説にあるとおり、 (num, tyy) は答えの型代入の一要素になり、また残りにその型代入を行ったリストを再帰的に評価するので、先ほど定義した map_subst を利用

して、unify (map_subst [(num, tyy)] rest) を返す。t1、t2 がそれぞれ tyy、TyVar num のときも同様である。t1、t2 がそれぞれ TyList ty1、TyList ty2 のときは、ty1 と ty2 が等しい必要があるので、それを rest に含めたリストを再帰的に評価すればよく、unify ((ty1, ty2) :: rest) を返せば良い。これで unify の実装は完了する。

2.4 Exercise 4.3.4

unify の第一引数のリストに含まれる (TyVar num, tyy) について、num が tyy に完全に含まれるとき、まずこれが必ず等しくならないことは明らかである。(方程式 a = a + b (ただし、b != 0) が成り立たないのと原理的に同じである。) よって、型代入に (num, tyy) を含めてしまうと、tyy の中の num を tyy で置き換える、という操作を型代入が起きるたびに繰り返され、また型代入ではエラーが起きることはないので、間違った結果が返ってくる。

2.5 Exercise 4.3.5

IfExp(exp1, exp2, exp3) について型推論の手続きを与えると以下のようになる。

1. Γ、$exp1$ を入力として型推論を行い、$S1$、$\tau 1$ を得る。 (2.1)
2. Γ、$exp2$ を入力として型推論を行い、$S2$、$\tau 2$ を得る。 (2.2)
3. Γ、$exp3$ を入力として型推論を行い、$S3$、$\tau 3$ を得る。 (2.3)
4. 型代入 $S1, S2, S3$ を$\alpha = \tau$という方程式の集まりとみなして、
   $S1 \cup S2 \cup S3 \cup (\tau 1, bool) \cup (\tau 2, \tau 3)$ を単一化し、型代入 $S4$ を得る。 (2.4)
5. $S4$ と$\tau 2$ を出力として返す。 (2.5)

次に LetInExp(id, exp1, exp2) についての型推論の手続きを与えると以下のようになる。

1. $\Gamma, exp1$ を入力として型推論を行い、$S1, \tau 1$ を得、
   id を$\tau 1$ でΓを拡張した新しい環境 $N\Gamma$を作る。 (2.6)
2. $N\Gamma, exp2$ を入力として型推論を行い、$S2, \tau 2$ を得る。 (2.7)
3. 型代入 $S1, S2,$ を$\alpha = \tau$という方程式の集まりとみなして、
   $S1 \cup S2 \cup$ を単一化し、型代入 $S3$ を得る。 (2.8)

4. $S3$ と$\tau 2$ を出力として返す。 (2.9)

次に FunExp(id, exp) についての型推論の手続きを与えると以下のようになる。

1. Γ を入力とし、新しい型変数$\tau 1$ を作り、Γを、
   $\tau 1$ に束縛された id で拡張した新しい環境 $N\Gamma$を作る。 (2.10)

2. $N\Gamma$、exp を入力とし、型推論を行い、$S1$, $\tau 2$ を得る。 (2.11)

3. $\tau 1$ に $S1$ の型代入を適用してできた新たな型、$\tau 3$ を得た後、
   $S1, TyFun(\tau 3, \tau 2)$ を返す。 (2.12)

次に、AppExp(exp1, exp2) についての型推論の手続きを与えると以下のようになる。

1. $\Gamma, exp1$ を入力とし、型推論を行い、$S1$, $\tau 1$ を得る。 (2.13)

2. $\Gamma, exp2$ を入力とし、型推論を行い、$S2, \tau 2$ を得る。 (2.14)

3. $\tau 1$ の型が $TyFun(tyy1, tyy2)$ のとき、手続き 4 へ、
   $TyVar\ num$ のときは手続き 5 へ
   どちらでもない時はエラー発生 (2.15)

4. $S1 \cup S2 \cup (tyy1, \tau 2)$ を単一化し、型代入 $S3$ を得、
   $S3, tyy2$ を出力として返す。 (2.16)

5. 新しい型変数$\tau 3$ を得、$S1 \cup S2 \cup (TyVarnum, TyFun(\tau 2, \tau 3))$
   を単一化し、型代入 $S4$ を得、$S4$, $\tau 3$ を出力として返す。 (2.17)

次に、補助関数について解説する。ひとつ目は、型代入を型の等式集合に変換する関数 eqs_of_subst である。これは、リストを走査しながら (num, ty) を (Tyvar num, ty) に全てを変換すれば良いだけである。また、解説 pdf では補助関数 subst_eqs を定義しているが、これは map_subst そのものである。次に、先ほどの型推論の実装であるが、基本的には手続き通りに実装すれば良い。ただし、単一化をする際の入力となるリストは、先ほどの補助関数 eqs_of_subst を利用して型代入のリストの型を変換してから unify に渡す必要があることに注意する。AppExp に関して少し補足すると、ty1 の形として許されるのは、TyFun、TyVar num の２つである。TyVar num についてだが、ty1 はこの段階で初めて関数型であることが決まり、最終的に返す型はまだ確定していないので、新しい型変数を作る必要がある。