PPL Assignment 3

Submitted by Daniel Baer Sachs - 318645058

Part 1: Theoretical Questions

Question 1.1

1:

(a) `[T1*[T1->T2]->N] , [[T3->T4]*[T5->Number]->N]`

MGU exists: `{T1 = [T3->T4], T2 = Number, T5 = [T3->T4]}`

(b) `[T1*[T1->T2]->N] , [Number*[Symbol->T4]->N]`

No MGU exists - No possible unifier because Number and Symbol are incompatible atomic types.

(c) `T1 , T2`

MGU exists: `{T1 = T2}`

(d) `Boolean , Boolean`

MGU exists: `{}`

2:

(a) `{f: [T2->T3], g: [T1->T2], a: Boolean} |- (f (g a)): T3`

- First, we look at `(g a)` :
  - `g` has type `[T1->T2]`
  - `a` has type `Boolean`
  - For the application `(g a)` to be well-typed, we need `T1 = Boolean`
  - Therefore, `(g a)` has type `T2`
- Next, We look at `(f (g a))` :

- `f` has type `[T2->T3]`
- `(g a)` has type `T2` (from before)
- The application `(f (g a))` is well-typed and has type `T3`

True - The typing statement is valid. With the constraint that `T1 = Boolean`, the expression `(f (g a))` has type `T3` as desired.

(b) `{f: [T2->T1], x: T1, y: T3} |- (f x): T1`

- `f` has type `[T2->T1]`
- `x` has type `T1`
- `y` has type `T3`
- For the application `(f x)` to be well-typed, the argument type must match the parameter type of `f`, `f` expects an argument of type `T2`, but `x` has type `T1`
- For this to work, we need `T2 = T1`
- If `T2 = T1`, then `f` has type `[T1->T1]`, and `(f x)` would have type `T1`

True - The typing statement is valid when we unify `T2 = T1`. Under this constraint, `f` has type `[T1->T1]`, `x` has type `T1`, and `(f x)` has type `T1` as desired.

Question 1.2

```
(lambda (x)
  (lambda (y)
    (lambda (z)
      (lambda (w)
        (lambda (v)
          (x (y (z (w v)))))))))
```