

LING419 Homework 1 Solutions

- (4) a. $((\lambda x \lambda y. x^2 + 3xy + 4)(1))(3)$
 $= (\lambda y. 1^2 + 3 \times 1 \times y + 4)(3)$
 $= 1^2 + 3 \times 1 \times 3 + 4$
- b. $(\lambda x \lambda y. x^2 + 3xy + 4)(2)$
 $= \lambda y. 2^2 + 3 \times 2 \times y + 4$
- c. $(\lambda y. met'(y)(john'))(mary')$
 $= met'(mary')(john')$
- d. $(\lambda f. f(3))(\lambda w. w + 2)$
 $= (\lambda w. w + 2)(3)$
 $= 3 + 2$
- e. $(\lambda f. f(3))((\lambda x \lambda y. x^2 + 3xy + 4)(1))$
 $= (\lambda f. f(3))(\lambda y. 1^2 + 3 \times 1 \times y + 4)$
 $= (\lambda y. 1^2 + 3 \times 1 \times y + 4)(3)$
 $= 1^2 + 3 \times 1 \times 3 + 4$
- f. $(\lambda f. f(mary'))(\lambda x. walks'(x))$
 $= (\lambda x. walks'(x))(mary')$
 $= walks'(mary')$

(5) $ApplyThenDouble' = \lambda f \lambda x \lambda y. 2 \times f(x)(y)$

- a. $(ApplyThenDouble'(\lambda a \lambda b. 3a + 2b))(5)(3)$
 $= ((\lambda f \lambda x \lambda y. 2 \times f(x)(y))(\lambda a \lambda b. 3a + 2b))(5)(3)$
 $= (\lambda x \lambda y. 2 \times (\lambda a \lambda b. 3a + 2b)(x)(y))(5)(3)$ (fill the f slot with $\lambda a \lambda b. 3a + 2b$)
 $= (\lambda x \lambda y. 2 \times (\lambda b. 3x + 2b)(y))(5)(3)$ (fill the a slot with x)
 $= (\lambda x \lambda y. 2 \times (3x + 2y))(5)(3)$ (fill the b slot with y)
 $= (\lambda y. 2 \times (3 \times 5 + 2y))(3)$ (fill the x slot with 5)
 $= 2 \times (3 \times 5 + 2 \times 3)$ (fill the y slot with 3)
- b. $(ApplyThenDouble'(\lambda p \lambda q. p - q))$
 $= ((\lambda f \lambda x \lambda y. 2 \times f(x)(y))(\lambda p \lambda q. p - q))$
 $= (\lambda x \lambda y. 2 \times (\lambda p \lambda q. p - q)(x)(y))$ (fill the f slot with $\lambda p \lambda q. p - q$)
 $= (\lambda x \lambda y. 2 \times (\lambda q. x - q)(y))$ (fill the p slot with x)
 $= \lambda x \lambda y. 2 \times (x - y)$ (fill the q slot with y)
- c. $(ApplyThenDouble'(\lambda w \lambda x. w + x))(2)(3)$
 $= ((\lambda f \lambda x \lambda y. 2 \times f(x)(y))(\lambda w \lambda x. w + x))(2)(3)$
 $= ((\lambda f \lambda x \lambda y. 2 \times f(x)(y))(\lambda w \lambda z. w + z))(2)(3)$ (rename x to z to avoid clashes)
 $= (\lambda x \lambda y. 2 \times (\lambda w \lambda z. w + z)(x)(y))(2)(3)$ (fill the f slot with $\lambda w \lambda z. w + z$)
 $= (\lambda x \lambda y. 2 \times (\lambda z. x + z)(y))(2)(3)$ (fill the w slot with x)
 $= (\lambda x \lambda y. 2 \times (x + y))(2)(3)$ (fill the z slot with y)
 $= (\lambda y. 2 \times (2 + y))(3)$ (fill the x slot with 2)
 $= 2 \times (2 + 3)$ (fill the y slot with 3)

If thinking about semantic types like e and t and $(e \rightarrow t)$ helps you understand what's going on sometimes, then the following might help you understand what's going on with this $ApplyThenDouble'$ function.

We've talked about semantic types like e and t and $(e \rightarrow t)$ for the semantic values of words and phrases. Let's say that there's also a type representing numbers, in the way that e represents entities and t represents

truth values. Let's call this new type n . So 0 and 1 and 2 are of type n ; and a function like $(\lambda x.x + 1)$ is of type $(n \rightarrow n)$, because you give it a number and it gives back another number. Functions like $(\lambda w \lambda x.w + x)$ and $(\lambda a \lambda b.3a + 2b)$ are of type $(n \rightarrow (n \rightarrow n))$, because you give them a number and they give you back a function which works like $(\lambda x.x + 1)$, i.e. a function of type $(n \rightarrow n)$.

Now, what is the type of *ApplyThenDouble'*? Well, it first takes a function like $(\lambda w \lambda x.w + x)$ or $(\lambda a \lambda b.3a + 2b)$; and then we give the resulting thing a number; and then we give the resulting thing a number. After all that, we get back a number. So the type of *ApplyThenDouble'* is:

$$(n \rightarrow (n \rightarrow n)) \rightarrow (n \rightarrow (n \rightarrow n))$$

This just says that we need to give it a function of type $(n \rightarrow (n \rightarrow n))$, then a number, and then a number, and then we'll get back a number.

But writing the type out like that tells us that we can also think of *ApplyThenDouble'* as a function that takes one argument — specifically an argument of type $(n \rightarrow (n \rightarrow n))$ — and then gives back another thing of exactly the same type. This way of thinking about things was particularly appropriate in 5b, where we applied *ApplyThenDouble'* to the function $\lambda p \lambda q.p - q$ and got back a new function of the same type as a result, namely $\lambda x \lambda y.2(x - y)$. Put differently, we gave *ApplyThenDouble'* the function that subtracts one number from another, and we got back as a result the function that subtracts one number from another and then doubles the result. Also notice that even in 5a, we really started by applying *ApplyThenDouble'* to $\lambda a \lambda b.3a + 2b$ to get $\lambda x \lambda y.2(3x + 2y)$, and then we applied that to 5 and to 3.

This might help make sense of the idea that if you have a function whose type looks like $a \rightarrow (b \rightarrow c)$, you can think of it in either of two ways:

- The official way is: we give it something of type a , and it will give back something of type $(b \rightarrow c)$.
- The unofficial (but sometimes friendlier) way is: we give it something of type a “and then” give it something of type b , and it will give back something of type c .