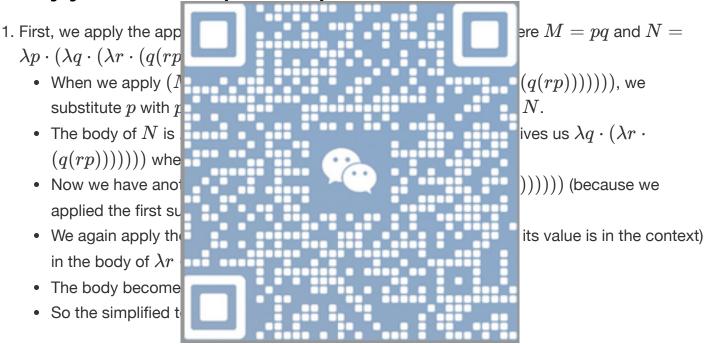
### COMP4161 T3/2024 Advanced Topics in Software Verification - Assignment 1

#### 1. λ-Calculus (16 marks)

(a) Simplify the term  $(pq)(\lambda p\cdot (\lambda q\cdot (\lambda r\cdot (q(rp))))))$  syntactically by applying the syntactic conventions and rules. Justify your answer. (2 marks)



# (b) Restore the omitted parentheses in the term $a(\lambda ab.(bc)a(bc))(\lambda b.cb)$ (but make sure you don't change the term structure). (2 marks)

- 1. The term should be  $a((\lambda ab.((bc)a(bc))))(\lambda b.cb))$ .
  - The innermost lambda expression  $\lambda ab.(bc)a(bc)$  needs its own parentheses around the body.
  - And the whole application of a to the lambda expression and then to  $(\lambda b.cb)$  also needs appropriate parentheses.

(c) Find the normal form of  $(\lambda f \cdot \lambda x \cdot f(fx))(\lambda g \cdot \lambda y \cdot g(g(gy)))$ ). Justify your answer by showing the reduction sequence. Each step in the reduction sequence should be a single  $\beta$ -reduction step. Underline or otherwise indicate the redex being reduced for each step. (6 marks)

- 1. Let's start with the term  $(\lambda f \cdot \lambda x \cdot f(fx))(\lambda g \cdot \lambda y \cdot g(g(gy))))$ .
  - The redex is  $(\lambda f \cdot \lambda x \cdot f(fx))(\lambda g \cdot \lambda y \cdot g(g(gy)))$ . We substitute f with  $\lambda g \cdot \lambda y \cdot g(g(gy))$  and x with any value (let's say z for simplicity, but it doesn't matter in this context as we are just reducing syntactically).
  - After the first  $\beta$ -reduction, we get  $\lambda x \cdot (\lambda g \cdot \lambda y \cdot g(g(gy)))(\lambda g \cdot \lambda y \cdot g(g(gy))x))$ .
  - Now the new redex is  $(\lambda a \cdot \lambda u \cdot a(a(au)))(\lambda a \cdot \lambda u \cdot a(a(au))x))$ . We substitute g with  $\lambda g \cdot \lambda y \cdot g(g(gy))$
  - After the second  $\beta$   $\lambda y \cdot g(g(gy))(\lambda g \cdot \lambda y \cdot g(g(gy)))(\lambda g \cdot \lambda y \cdot g(g(gy))y)$
  - We can continue t will keep expanding and we won't reach a recause the function  $\lambda g \cdot \lambda y \cdot g(g(gy))$  is

ambda calculus

### (d) Recall the end (Church Numeral

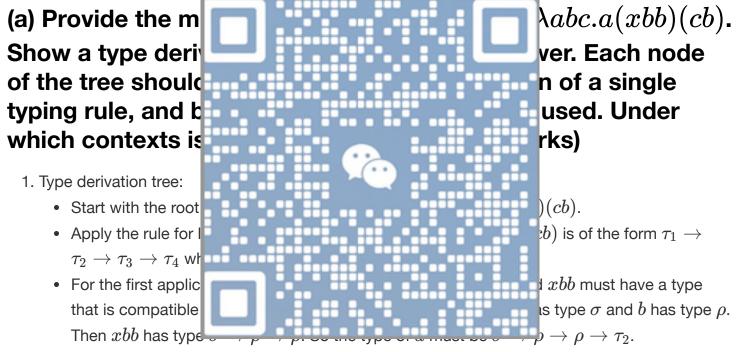
- $0 \equiv \lambda f x, x$
- $1 \equiv \lambda f x. f x$
- $2 \equiv \lambda f x. f(f x)$
- $3 \equiv \lambda f x. f(f(fx))...$

Define exp where exp m n beta-reduces to the Church Numeral representing  $m^n$ . Provide a justification of your answer. (6 marks)

- 1. We want to define an exp function in lambda calculus. Let's first consider what the operation should do.
  - If we have m and n as Church Numerals, we want to apply the function m times to another function, and then apply that result n times to the argument x.
- 2. We define exp as follows:
  - $\exp \equiv \lambda mnfx.m(nf)x$
- 3. Justification:
  - Let  $m=\lambda f_m x_m.f_m^m(x_m)$  and  $n=\lambda f_n x_n.f_n^n(x_n)$  (where  $f_m^m(x_m)$  means applying  $f_m$  to  $x_m$  m times and similarly for n).

- First, we consider the inner application nf.
  - Substituting f with f and x with f in the body of n, we get  $nf = \lambda x_n.f_n^n(f)$ .
- Then we apply m to (nf).
  - Substituting f with (nf) and x with x in the body of m, we get  $m(nf)=\lambda x_m.(nf)^m(x_m)=\lambda x_m.(\lambda x_n.f_n^n(f))^m(x_m).$
  - $\circ$  This means applying the function nf (which is applying  $f_n^n(f)$ ) m times to  $x_m$ .
- Finally, we apply the result to x.
  - $\circ$  So exp m n f x will result in applying the function m times to nf and then applying that result to x, which is equivalent to the Church Numeral representation of  $m^n$ .

#### 2. Types (20 marks)



- For the second application (cb), c has type  $au_3$  and b has type ho. So  $au_3$  must be  $ho o au_4$ .
- Combining these, the most general type is  $(\sigma \to \rho \to \rho \to \tau_2) \to (\rho \to \tau_4) \to \tau_1 \to \tau_2 \to \tau_3 \to \tau_4$ .
- The term is type correct under the context where the types of variables are assigned as described above (i.e.,  $x:\sigma,b:\rho,a:\sigma\to\rho\to\rho\to\tau_2,c:\rho\to\tau_4$ ).

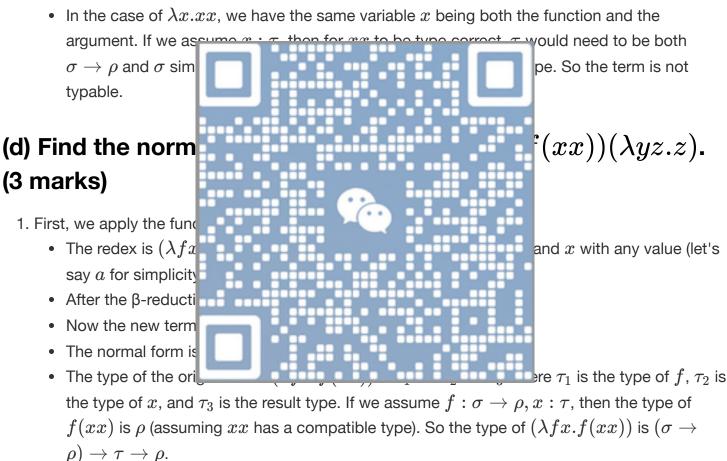
#### (b) Find a closed lambda term that has the following type:

- $('a\Rightarrow'b)\Rightarrow('c\Rightarrow'a)\Rightarrow'c\Rightarrow'b$ (You don't need to provide a type derivation, just the term). (4 marks)
- 1. The term  $\lambda fgx.f(gx)$  has the given type.

- Let's assume f has type  $(a \Rightarrow b)$ , g has type  $(c \Rightarrow a)$ , and x has type c.
- Then gx has type a (by applying the function g of type  $c \Rightarrow a$  to a of type c.
- Then f(gx) has type 'b (by applying the function f of type  $(a \Rightarrow b)$  to gx of type 'a).
- So  $\lambda fgx.f(gx)$  has the type  $('a \Rightarrow' b) \Rightarrow ('c \Rightarrow' a) \Rightarrow' c \Rightarrow' b$ .

#### (c) Explain why $\lambda x.xx$ is not typable. (3 marks)

- 1. The term  $\lambda x.xx$  is not typable because when we try to assign a type to it, we run into a problem.
  - Let's assume x has type  $\tau$ . Then the term xx is applying the function x to itself. But for a function application to be type correct, the type of the function (the first x) must be of the form  $\sigma \to \rho$  and the type of the argument (the second x) must be  $\sigma$ .



• The term  $(\lambda yz.z)$  has type  $\sigma \to \tau \to \tau$ . When we apply the first term to the second term, the resulting type is  $\tau \to \tau$ . So the type of the normal form  $\lambda x.z$  is  $\tau \to \tau$ .

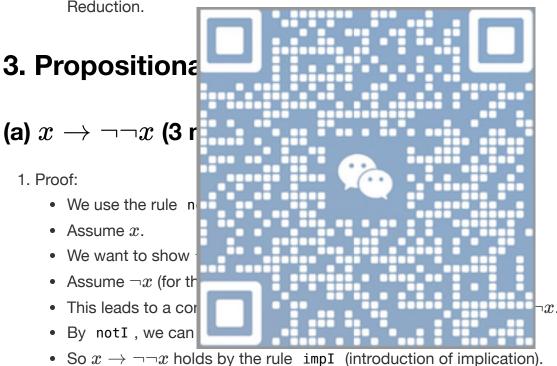
## (e) Is $(\lambda fx.f(xx))(\lambda yz.z)$ typable? Compare this situation with the Subject Reduction that you learned in the lecture. (5 marks)

1. The term  $(\lambda fx.f(xx))(\lambda yz.z)$  is typable.

• As we saw in part (d), the type of  $(\lambda fx.f(xx))$  is  $(\sigma \to \rho) \to \tau \to \rho$  and the type of  $(\lambda yz.z)$  is  $\sigma \to \tau \to \tau$ . The types are compatible for application.

#### 2. Regarding Subject Reduction:

- Subject Reduction states that if a term M has a type  $\tau$  and M reduces to N (i.e.,  $M \to N$  ), then N also has a type and the type is related to  $\tau$ .
- In this case, we started with  $(\lambda fx.f(xx))(\lambda yz.z)$  and reduced it to  $\lambda x.z$ . The original term had a certain type as we determined, and the reduced term also has a type.
- The type of the reduced term is a consequence of the reduction and the typing rules. The
  fact that the term can be typed before and after reduction is consistent with the idea of
  Subject Reduction. If the term was not typable before reduction and became typable after
  reduction (or vice versa), it would violate the principle of Subject Reduction. Here, the typing
  is consistent throughout the reduction process, which aligns with the concept of Subject
  Reduction



#### (b) (X o Y o eg X) o X o eg Y (3 marks)

#### 1. Proof:

- Assume (X o Y o 
  eg X).
- Also assume X.
- We want to show  $\neg Y$ .
- From  $(X \to Y \to \neg X)$  and X, by impE (elimination of implication), we get  $Y \to \neg X$ .
- Since we have X and  $Y \to \neg X$ , and assuming Y (for the purpose of <code>notI</code> ), we get  $\neg X$  by <code>impE</code> .
- ullet But we already have X, so this is a contradiction.