UNIVERSITY OF BRISTOL

January 2021 Examination Period

FACULTY OF ENGINEERING

Third Year Examination for the Degrees
of
Bachelor of Science
Master of Engineering

COMS30039J
Types and Lambda Calculus

TIME ALLOWED:
2 Hours

Answers to COMS30039J: Types and Lambda Calculus

Intended Learning Outcomes:

Q1. This question concerns the pure, untyped λ-calculus.

It will be useful to recall the combinators $\underline{\text{id}} = \lambda x.\, x$ and $\underline{\text{const}} = \lambda xy.\, x$

(a) Give the set of free variables for each of the following terms:

i. $\lambda xyz.\, xz(yz)$
ii. $z(\lambda x.\, xy)$
iii. $\lambda z.\, (\lambda x.\, y)(xz)$
iv. $(\lambda x.\, x\,(\lambda y.\, xy))(\lambda x.\, y)$

[4 marks]

Solution:

i. $\emptyset$
ii. $\{z, y\}$
iii. $\{x, y\}$
iv. $\{y\}$

(b) For each of the following, give a term M that satisfies the statement.

i. $M\,x \;\approx\; xx$
ii. $M\,\underline{n} \;\approx\; \underline{2 * n}$
iii. $M \;\approx\; MM$
iv. $M\,x \;\approx\; MM$

[8 marks]

Solution: Two marks each.

i. $\lambda y.\, yy$
ii. $\lambda x.\, \underline{\text{add}}\ x\ x$
iii. $\lambda x.\, x$
iv. $\lambda x.\, y$

(c) Define combinator $\underline{\text{sub}}$ as follows:

$$\underline{\text{sub}} \quad := \quad \text{fix}\,(\lambda f\,mn.\, \text{ifz}\,n\,m\,(f\ (\text{pred}\ m)\ (\text{pred}\ n)))$$

Prove, by induction on n, that $\underline{\text{sub}}$ satisfies:

$$\underline{\text{sub}}\ \underline{m}\ \underline{n} \;\approx\; \begin{cases} \underline{0} & \text{if } m \leq n \\ \underline{m - n} & \text{otherwise} \end{cases}$$

[6 marks]

Solution: 2 marks for correct form of induction proof. 2 marks for any appropriate case splitting within step case. 2 marks for generally correct reasoning.

We will use the following fact, which is true simply by evaluating fix:

$$\underline{\mathsf{sub}} \;\approx\; \lambda mn.\, \mathsf{ifz}\, n\, m\, (\underline{\mathsf{sub}}\, (\mathsf{pred}\, m)\, (\mathsf{pred}\, n))$$

The proof is by induction on n.

- When $n = 0$, by reduction $\underline{\mathsf{sub}}\ \underline{m}\ \underline{0} \;\approx\; \underline{m}$.
- When $n = k+1$ for some k, we have:

$$\begin{array}{ll} & \underline{\mathsf{sub}}\ \underline{m}\ \underline{(k+1)} \\ \approx & \underline{\mathsf{sub}}\, (\mathsf{pred}\, \underline{m})\, (\mathsf{pred}\, \underline{(k+1)}) \\ \approx & \underline{\mathsf{sub}}\, \underline{(m \ominus 1)}\, \underline{k} \end{array}$$

  where $m \ominus 1$ is 0 if m is 0 and is $m-1$ otherwise. Then we distinguish cases on whether or not $m \leq k+1$:

  - If $m \leq k+1$ then $m \ominus 1 \leq k$ too. Hence, it follows from the induction hypothesis that the last line above is convertible with $\underline{0}$, which is correct in this case since $m \leq n$.
  - Otherwise $m \leq k+1$ and so $m - 1 > k$ too. Hence, it follows from the induction hypothesis that the last line above is convertible with $\underline{m-1-k}$. This is correct since $m - (k+1) = m - k - 1$.

(d) Prove that there does not exist a term M such that, for all terms N:

$$M\, N \;\approx\; \begin{cases} \underline{\mathsf{id}} & \text{if } N \text{ is in normal form} \\ \underline{\mathsf{const}} & \text{otherwise} \end{cases}$$

[3 marks]

Solution: Suppose for contradiction that such an M exists. Then we would have:

$$\underline{\mathsf{id}} \;\approx\; M\ \underline{\mathsf{id}} \;\approx\; M\ (\underline{\mathsf{id}}\ \underline{\mathsf{id}}) \;\approx\; \underline{\mathsf{const}}$$

but this is impossible since $\underline{\mathsf{id}}$ and $\underline{\mathsf{const}}$ are distinct normal forms.

(e) Find a finite sequence of *closed* terms $M_1, M_2, \ldots, M_k$ for $k \geq 0$ such that the following two equations are both satisfied:

$$\begin{array}{rcl} (\lambda x.\, x\, \underline{\mathsf{id}}\, (x\, \underline{\mathsf{id}}\, \underline{\mathsf{id}}))\, M_1\, M_2 \cdots M_k\, x\, y & \approx & x \\ (\lambda x.\, x\, \underline{\mathsf{id}}\, (x\, x\, \underline{\mathsf{id}}))\, M_1\, M_2 \cdots M_k\, x\, y & \approx & y \end{array}$$

[3 marks]

(cont.)

Solution: This is extremely difficult. An appropriate sequence is, $k = 6$:

$$(\lambda xyz.\, zxy), (\lambda xy.\, y), (\lambda xy.\, x), (\lambda wxyz.\, y), (\lambda x.\, x), (\lambda wxyz.\, z)$$

Q2. This question concerns type systems.

(a) Give a typing derivation for each of the following judgements:

i. $\vdash \lambda xy.\, x : a \to b \to a$

ii. $\vdash \lambda x.\, x(\lambda y.\, y) : ((a \to a) \to b) \to b$

iii. $y : c \vdash (\lambda x.\, y)(\lambda xz.\, x) : c$

[6 marks]

Solution: 2 marks each.

i.

$$\dfrac{\dfrac{\dfrac{}{x : a, y : b \vdash x : a}}{x : a \vdash \lambda y.\, x : b \to a}}{\vdash \lambda xy.\, x : a \to b \to a}$$

ii.

$$\dfrac{\dfrac{\dfrac{}{x : (a \to a) \to b \vdash x : (a \to a) \to b} \qquad \dfrac{\dfrac{}{x : (a \to a) \to b, y : a \vdash y : a}}{x : (a \to a) \to b \vdash \lambda y.\, y : a \to a}}{x : (a \to a) \to b \vdash x(\lambda y.\, y) : b}}{\vdash \lambda x.\, x(\lambda y.\, y) : ((a \to a) \to b) \to b}$$

iii.

$$\dfrac{\dfrac{\dfrac{}{y : c, x : a \to b \to a \vdash y : c}}{y : c \vdash \lambda x.\, y : (a \to b \to a) \to c} \qquad \dfrac{\dfrac{\dfrac{}{y : c, x : a, z : b \vdash x : a}}{y : c, x : a \vdash \lambda z.\, x : b \to a}}{y : c \vdash \lambda xz.\, x : a \to b \to a}}{y : c \vdash (\lambda x.\, y)(\lambda xz.\, x) : c}$$

(b) For each of the following types, find a closed pure term that inhabits the type:

i. $a \to b \to b$

ii. $(a \to b) \to (a \to b \to c) \to a \to c$

iii. $c \to ((a \to b \to a) \to c \to d) \to d$

[6 marks]

Solution: 2 marks each.

i. $\lambda xy.\, y$

ii. $\lambda xyz.\, y\, z\, (xz)$

iii. $\lambda xy.\, y\, (\lambda xy.\, x)\, x$

(c) Prove, by induction M, that: for all M, A, Γ, Γ', if $\Gamma \vdash M : A$ and $\Gamma \subseteq \Gamma'$ then $\Gamma' \vdash M : A$.

(cont.)

[7 marks]

Solution: One mark for proof of correct shape, one mark for correctly identifying induction hypotheses, one mark per case and an additional mark for constructing Γ' correctly to make use of the induction hypothesis in the abs case.
The proof is by induction on $\Gamma \vdash M : A$.

- When M is a variable x, let A be a type, Γ and Γ' be type environments such that $\Gamma \subseteq \Gamma'$ and suppose $\Gamma \vdash x : A$. By inversion, it follows that $x : A \in \Gamma$. Since Γ' contains all the typings of Γ, also $x : A \in \Gamma'$. Hence, by (TVar), $\Gamma' \vdash x : A$.

- When M is a constant c, let A be a type, Γ and Γ' be type environments such that $\Gamma \subseteq \Gamma'$ and suppose $\Gamma \vdash c : A$. By inversion, it follows that $c{:}A \in \mathbb{C}$. Therefore, the side condition is fulfilled to use (TCst) to also justify $\Gamma' \vdash c : A$ (this rule does not place any requirements on the environment).

- When M is an application PQ, assume the induction hypotheses:

  (IH1) For all Γ'' and Γ''' and A', if $\Gamma'' \subseteq \Gamma'''$ and $\Gamma'' \vdash P : A'$ then $\Gamma''' \vdash P : A'$.

  (IH2) For all Γ'' and Γ''' and A', if $\Gamma'' \subseteq \Gamma'''$ and $\Gamma'' \vdash Q : A'$ then $\Gamma''' \vdash Q : A'$.

  Let A be a type, Γ and Γ' be environments such that $\Gamma \subseteq \Gamma'$. Then suppose $\Gamma \vdash PQ : A$. By inversion, there must be a type B such that $\Gamma \vdash P : B \to A$ and $\Gamma \vdash Q : B$. It follows from (IH1) with $\Gamma'' := \Gamma$ and $\Gamma''' := \Gamma'$ and $A' := B \to A$ that $\Gamma' \vdash P : B \to A$. It follows from (IH2) with $\Gamma'' := \Gamma$, $\Gamma''' := \Gamma'$ and $A' := B$ that $\Gamma' \vdash Q : B$. Therefore, by (TApp), $\Gamma' \vdash PQ : A$.

- When M is an abstraction $\lambda x.\, P$, assume the induction hypothesis:

  (IH) For all Γ'' and Γ''' and A', if $\Gamma'' \subseteq \Gamma'''$ and $\Gamma'' \vdash P : A'$ then $\Gamma''' \vdash P : A'$.

  Let A be a type, Γ and Γ' be type environments such that $\Gamma \subseteq \Gamma'$. Then suppose $\Gamma \vdash \lambda x.\, P : A$. By the variable convention we can assume that x does not occur in Γ or Γ'. By inversion, it follows that there are types B and C such that $A = B \to C$ and $\Gamma, x{:}B \vdash P : C$. Then, it follows from the induction hypothesis with $\Gamma'' = \Gamma \cup \{x{:}B\}$ and $\Gamma''' = \Gamma' \cup \{x : B\}$ and $A' := C$, that $\Gamma', x{:}B \vdash P : C$. It follows by (TAbs) that, therefore, $\Gamma' \vdash \lambda x.\, P : B \to C$, as required.

(d) Prove that the only *pure*, closed, normal form of type $a \to b \to a$ is $\lambda xy.\, x$.

[7 marks]

Solution: We exclude the possibility that it can be a variable, a constant or an application.

- The term cannot just be a variable, because it is closed.
- The term cannot just be a constant, because it is pure.
- The term cannot be an application: observe that any application PQ can be written $((F\,M_1)\ldots M_p)\;Q$ with F either an abstraction, a variable or a constant by repeatedly unfolding (or "looking inside") the left term in the application, starting with P, until eventually the left term is not itself an application. However, F cannot be any of these things. It cannot be an abstraction (the term is in normal form), it cannot be a naked variable (the term is closed) and it cannot be a constant (the term is pure). Hence, it must be that M cannot be an application at all.

So the term must be of shape $\lambda x.\,N$ for some N and inversion tells us that N has type $b \to a$ under assumption $x : a$. We argue similarly that N cannot be a variable, a constant or an application.

- N cannot be a variable, because, by inversion, it would have to be x of type a, but $a \neq b \to a$.
- N cannot be a constant because the term is pure.
- N cannot be an application since, by the reasoning above, the term F in head position could only be a variable and then could only be x which would be excluded by inversion (x would have to have type $B \to b \to a$ for some B).

Hence, N is of shape $\lambda y.\,P$ and inversion tells us that $P : a$. Finally, we argue that P must be x. Since P cannot be an abstraction due to inversion (a is not an arrow), it must be of shape $F\ M_1 \cdots M_k$ for some k which is possibly 0 (i.e. no arguments) and some F which is not itself an application. We will argue that F must be a variable.

- F cannot be an abstraction: if $k \geq 1$ then that would mean the term is not in normal form, and if $k = 0$ then P would be an abstraction, which we already excluded.
- P cannot be a constant since it is pure.

Hence, F must be a variable. Since the term is closed, by inversion, it can only be that $F = x$ or $F = y$. In both cases, inversion gives us that k must be 0 (otherwise F would need to have an arrow type). So P is just F and hence $x{:}a,\ y{:}b \vdash F : a$. By inversion, it must be that $F = x$.

END OF PAPER