Harry Langford
hjel2@cam.ac.uk

1 Question 1

(a) Let M be the λ-term $\lambda xy.x(\lambda z.zu)y$. What is the β-normal form of the term $N = M(\lambda vw.v(wb))(\lambda xy.yaz)$?

$$
\begin{aligned}
N =& M(\lambda vw.v(wb))(\lambda xy.yaz) \\
=& (\lambda xy.x(\lambda z.zu)y)(\lambda vw.v(wb))(\lambda xy.yaz) \\
\twoheadrightarrow & (\lambda y.(\lambda vw.v(wb))(\lambda z.zu)y)(\lambda xy.yaz) \\
\twoheadrightarrow & (\lambda y.(\lambda w.v(wb))[\lambda z.zu/v]y)(\lambda xy.yaz) \\
=& (\lambda y.(\lambda w.(\lambda z.zu)(wb))y)(\lambda xy.yaz) \\
\twoheadrightarrow & (\lambda y.(\lambda w.wbu)y)(\lambda xy.yaz) \\
\twoheadrightarrow & (\lambda w.wbu)(\lambda xy.yaz) \\
\twoheadrightarrow & (\lambda xy.yaz)bu \\
\twoheadrightarrow & (\lambda y.yaz)u \\
\twoheadrightarrow & uaz \\
\not\twoheadrightarrow &
\end{aligned}
$$

So the $\beta - NF$ of N is uaz

(b) Apply the simultaneous substitution $\sigma = [x/y, (\lambda xy.zy)/u]$ to M and N and find the β-normal form of $N[\sigma]$

$$
\begin{aligned}
M[\sigma] &= (\lambda xy.x(\lambda z.zu)y)[x/y, (\lambda xy.zy)/u] \\
&= (\lambda xy.x(\lambda w.wu)y)[(\lambda xy.zy)/u] \\
&= (\lambda xy.x(\lambda w.w(\lambda xy.zy))y) \\
N[\sigma] &= uaz[x/y, (\lambda xy.zy)/u] \\
&= (\lambda xy.zy)az \\
&\twoheadrightarrow (zy)[a/x.z/y] \\
&= zz
\end{aligned}
$$

(c) Give 2 terms α-equivalent to M. Give 2 other terms β-equivalent to M.

$$
\begin{aligned}
M &=_\alpha \lambda yz.y(\lambda z.zu)z =_\alpha \lambda xy.x(\lambda x.xu)y \\
M &=_\beta (\lambda fxy.xfy)(\lambda z.zu) =_\beta (\lambda x.x)(\lambda xy.x(\lambda z.zu)y)
\end{aligned}
$$

Can prepend λx.x to the beginning of terms to get an infinite series of terms.

(d) We define η-equivalence as: $M =_\eta \lambda x.Mx$ for any λ-term M. Give a shorter and a longer term η-equivalent to M. η-equivence is preserved under substitution

"equivalence" implies we also have the two equivalence rules

$$\frac{M = M' \quad N = N'}{M\,N = M'\,N'} \qquad \frac{M = M'}{\lambda x.\, M = \lambda x.\, M'}$$

Note the brackets are important! Else this gets sketchy and isn't valid... IE the domain of the outermost λ parameter is too much!

$$M =_\eta M$$

$$M =_\eta \lambda xyz.((Mx)y)z$$

can be simpler

$M =_\eta \lambda z.(\lambda y.(\lambda x.\, M x)\, y)\, z$

(e) What is the use of η-equivalence in functional programming?

This enables compiler optimisations. At any point, we can replace an expression with any expression which is η-equivalent to it.

"Demonstrates the correctness of some" curried functions

Only for functions (not values)

2 Question 2

(a) Give a complete proof of the correctness of Church addition from Slide 119.

$$Plus(m, n) =_\beta \langle m + n \rangle$$

We are required to prove that for all church numerals $\underline{n}$, $\underline{m}$ the following holds:

$$\underline{m+n} =_\beta \lambda\ f\ x.\underline{m}\ f\ (\underline{n}\ f\ x)$$

I will prove this by assuming arbitrary n and perform mathematical induction over m. Since the proof assumes n is arbitrary, it holds for all n. By induction over m, we will prove it holds for all m.

- Case $m = 0$

$$\begin{aligned}
&\lambda\ f\ x.\ \underline{0}\ f\ (\underline{n}\ f\ x)\\
=_\beta&\lambda\ f\ x.\ (\lambda\ f\ x.\ x)\ f\ (\underline{n}\ f\ x)\\
=_\beta&\lambda\ f\ x.\ \underline{n}\ f\ x\\
=_\beta&\underline{n}\\
=_\beta&\underline{n+0}
\end{aligned}$$

  Therefore, the proof holds in the case $m = 0$.

- The inductive step.

  Assume the lemma holds for $m = k$ and prove it holds for $m = k + 1$:

$$\begin{aligned}
&\lambda\ f\ x.\ \underline{k+1}\ f\ (\underline{n}\ f\ x)\\
=_\beta&\lambda\ f\ x.\ \underline{k+1}\ f\ (f^n\ x)\\
=_\beta&\lambda\ f\ x.\ f(f^k(f^n\ x))\\
=_\beta&\lambda\ f\ x.\ f(\underline{n+k}\ f\ x)\\
=_\beta&\ Succ\ \ \underline{n+k}\\
=_\beta&\underline{n+k+1}
\end{aligned}$$

interesting

  Therefore, if the lemma holds for $m = k$ then it also holds for $m = k+1$. Since it held for $m = 0$, by mathematical induction, we have proved it holds for all $m \in \mathbb{N}$. Since n was arbitrary in the proof, this statement holds for all n. Therefore:

$$\forall n \forall m. \underline{m+n} =_\beta \lambda\ f\ x.\underline{m}\ f\ (\underline{n}\ f\ x)$$

  As required.

(b) Define the λ-terms Times and Exp representing multiplication and exponentiation of Church numerals respectively. Prove the correctness of your definitions.

$$\begin{aligned}
Times(m,n) &=_\beta \langle m * n\rangle\\
Exp(m,n) &=_\beta \langle m^n\rangle
\end{aligned}$$

Can exploit the properties of church numerals
Times(m, n) = λf. m (n f)
Exp(m, n) = m n

$$\begin{aligned}
Times(m,n) &= n\ (Add\ \ m)\ \underline{0}\\
Exp(m,n) &= n\ (Times\ \ m)\ \underline{1}
\end{aligned}$$

Be careful with the distinction between numbers and church numerals!

$$\begin{aligned}
Times(m,n) &= m\ (Add\ \ n)\ 0\\
&= \underbrace{m + \cdots + m}_{n\text{ times}} + 0 \text{ using correctness of } Add\\
&= \langle m \times n\rangle \text{ as required}
\end{aligned}$$

$$\begin{aligned}
Exp(m,n) &= m\ (Times\ \ n)\ 1\\
&= \underbrace{m \times \cdots \times m}_{n\text{ times}} \times 1 \text{ using correctness of } Times\\
&= \langle m^n\rangle \text{ as required}
\end{aligned}$$

3 Question 3

1. If you are *still* not fed up with Ackermann's function $ack \in \mathbb{N}^2 \to \mathbb{N}$, show that the λ-term $ack \triangleq \lambda x.x(\lambda fy.yf(f\underline{1}))\,Succ$ represents ack (where $Succ$ is as on slide 123).

   The Ackermann function has three rules:

$$\begin{aligned} ack(0, x_2) &\triangleq x_2 + 1 \\ ack(x_1 + 1, 0) &\triangleq ack(x_1, 1) \\ ack(x_1 + 1, x_2 + 1) &\triangleq ack(x_1, ack(x_1 + 1, x_2)) \end{aligned}$$

   To prove that the λ-term above is computes Ackermann's function, I will perform case analysis.

   - Case $ack(0, x_2)$

$$\begin{aligned} (\lambda x.x(\lambda fy.yf(f\underline{1}))\,Succ)\underline{0}\ \underline{x_2} &=_\beta (\underline{0}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{x_2} \\ &=_\beta Succ\ \ \underline{x_2} \\ &=_\beta \underline{x_2+1} \\ &=_\beta ack(0, x_2) \end{aligned}$$

   - Case $ack(x_1 + 1, 0)$ I will prove this by induction. Start by proving the base case $x_1 = 0$:

$$\begin{aligned} &(\lambda x.x(\lambda fy.yf(f\underline{1}))\,Succ)\underline{1}\ \underline{0} \\ \twoheadrightarrow &(\underline{1}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{0} \\ \twoheadrightarrow &(\lambda fy.yf(f\underline{1})\,Succ)\underline{0} \\ \twoheadrightarrow &(\lambda y.y\,Succ(Succ\,\underline{1}))\underline{0} \\ \twoheadrightarrow &\underline{0}\,Succ(Succ\,\underline{1}) \\ \twoheadrightarrow &\,Succ\,\underline{1} \\ \twoheadrightarrow &\underline{2} \end{aligned}$$

     Since $Ack(1, 0) = 2$, this proves that the λ-term ack computes Ackermann's function in the case $x_1 = 0$, $x_2 = 0$.

     Consider now the inductive step. Assume that the λ-term ack computes Ackermann's function for $x_1 = k$ and prove it computes the function for $k + 1$:

$$\begin{aligned} &(\lambda x.x(\lambda fy.yf(f\underline{1}))\,Succ)\underline{k+1}\ \underline{0} \\ \twoheadrightarrow &\underline{k+1}(\lambda fy.yf(f\underline{1}))\,Succ\,\underline{0} \\ \twoheadrightarrow &(\lambda fy.yf(f\underline{1}))(\underline{k}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{0} \\ \twoheadrightarrow &\underline{0}(\underline{k}(\lambda fy.yf(f\underline{1}))\,Succ)((\underline{k}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{1}) \\ \twoheadrightarrow &((\underline{k}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{1}) \end{aligned}$$

     summarize at the end

- Case $ack(x_1 + 1, x_2 + 1)$ induction?

$$
\begin{aligned}
&(\lambda x.x(\lambda fy.yf(f\underline{1}))\,Succ)\ \underline{x_1+1}\ \underline{x_2+1} \\
=_\beta\ &\underline{x_1+1}(\lambda fy.yf(f\underline{1}))\,Succ\ \ \underline{x_2+1} \\
=_\beta\ &(\lambda fy.yf(f\underline{1}))(\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\ \underline{x_2+1} \\
=_\beta\ &\underline{x_2+1}\ (\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\ ((\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{1}) \\
=_\beta\ &(\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)(\underline{x_2}\ (\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\ ((\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\underline{1})) \\
=_\beta\ &(\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\ \underbrace{(\lambda fy.yf(f\underline{1})\underline{x_1+1}\ \underline{x_2})}_{=_\beta\ ack(x_1+1,x_2)} \\
=_\beta\ &(\underline{x_1}(\lambda fy.yf(f\underline{1}))\,Succ)\ \underline{ack(x_1+1,x_2)} \\
=_\beta\ &(\lambda yf.yf(f\underline{1}))\underline{x_1}\ \ \underline{ack(x_1+1,x_2)} \\
=_\beta\ &\underline{ack(x_1, ack(x_1+1,x_2))}
\end{aligned}
$$

induction?

$\lambda x.x(\quad)\ \underline{x_2}$

be careful: are you talking about numbers or encodings? terms or mathematical functions?

λ-terms

Therefore, the λ-term computes Ackermann's function for

2. Let I be the λ-term $\lambda x.x$. Show that $\underline{n}\ I =_\beta I$ holds for every Church numeral $\underline{n}$.

$$
\begin{aligned}
\underline{n}\ I &= (\lambda fx.f^n x)I \\
&= \lambda x.I^n x \\
&= \lambda x.x \\
&= I
\end{aligned}
$$

Now consider

$$B \triangleq \lambda\ f\ g\ x.\ g\ x\ I\ (f\ (g\ x))$$

Assuming the fact about normal order reduction mentioned on slide 115, show that if partial functions $f, g \in \mathbb{N} \rightharpoonup \mathbb{N}$ are represented by closed λ-terms F and G respectively, then their composition $(f \circ g)(x) \equiv f(g(x))$ is represented by $B\ F\ G$.

B is required to ensure that g is defined. I provide a proof by case analysis below:

- Case $g(x)\uparrow$

$$
\begin{aligned}
(B\ F\ G)x &\triangleq (\lambda\ f\ g\ x.\ g\ x\ I\ (f\ (g\ x)))\ F\ G\ x \\
&\twoheadrightarrow G\ x\ I\ (F\ (G\ x))
\end{aligned}
$$

By assumption, $g(x)\uparrow$. So the λ-term $G\ x$ contains an infinite reduction sequence and has no normal form. Using the fact on slide 115, since $G\ x$ is the left-most λ-term we can conclude that the λ-term contains no normal form. Therefore, in this case of $(B\ F\ G)\ x$ undefined – as required.

- Case $g(x)\downarrow \wedge f(g(x))\downarrow$

$$
\begin{aligned}
(B\ F\ G)x &\triangleq (\lambda\ f\ g\ x.\ g\ x\ I\ (f\ (g\ x)))\ F\ G\ x \\
&\twoheadrightarrow G\ x\ I\ (F\ (G\ x)) \\
&\twoheadrightarrow \underline{g(x)}\ I\ (F\ (G\ x)) && \text{since } g \subseteq \mathbb{N}\times\mathbb{N} \\
&\twoheadrightarrow (F\ (G\ x)) && \text{using } \underline{n}\ I =_\beta I \\
&\twoheadrightarrow F\ \underline{g(x)} \\
&\twoheadrightarrow \underline{f(g(x))}
\end{aligned}
$$

what if f is undefined?

Case $g(x)\downarrow \wedge f(g(x))\uparrow$: Similar but $F\ \underline{g(x)}$ reduces infinitely

4 Question 4

In this question you may use all of the λ-definable functions presented in the notes, as well as the terms you define as part of this exercise. You should explain your answers (possibly using some examples), but don't need to prove their correctness.

Harry Langford
hjel2@cam.ac.uk

(a) "Not", i.e boolean negation

$$Not(f) \triangleq \lambda\ x\ y.f\ y\ x$$

Not inverts the result – in λ calculus this is achieved by swapping the order in which the λ-terms are passed to the boolean.

(b) "And" and "Or", i.e boolean conjunction and disjunction

$$And(f, g) \triangleq \lambda\ x\ y.f\ (g\ x\ y)\ y$$
$$Or(f, g) \triangleq \lambda\ x\ y.f\ x\ (g\ x\ y)$$

(c) "Minus", i.e truncated subtraction

$$Minus \triangleq \lambda\ n\ m.\ m\ (Pred)\ n$$

$Pred$ performs truncated subtraction for 1. So applying $Pred$ m times has the same effect as truncated subtraction of m from n.

(d) Numeric comparison operators $=, \neq, <, \leq, >, \geq$. You can define them in any order you find most convenient.

$$\geq \triangleq \lambda\ m\ n.\ Eq_0(Minus\ \ n\ m)$$
$$< \triangleq \lambda\ m\ n.\ Not(\geq m\ n)$$
$$\leq \triangleq \lambda\ m\ n.\ Eq_0(Minus\ \ m\ n)$$
$$> \triangleq \lambda\ m\ n.\ Not(\leq m\ n)$$
$$= \triangleq \lambda\ m\ n.\ And(\geq m\ n)(\leq m\ n)$$
$$\neq \triangleq \lambda\ m\ n.\ Not(= m\ n)$$

$m \geq n$ is the same as testing whether the truncated subtraction of m from n is equal to zero. So we do this.

$m < n$ is equivalent to $\neg(m \geq n)$, so I use this equivalence.

Similar logic applies to $\leq$ and $>$.

$=$ is equivalent to $(m \geq n) \wedge (m \leq n)$.

$m \neq n$ is logically equivalent to $\neg(m = n)$.

(e) A λ-term "MapPair", i.e a function that applies a function to both elements of the pair.

$$MapPair \triangleq \lambda\ f\ p.\ Pair\ (f\ (fst\ p))(f\ (snd\ p))$$

This destructs the pair, applies the function f to both halves and reconstructs the pair. This has the desired effect of "MapPair".

(f) "SquareSum", i.e a function representing $(m, n) \mapsto m^2 + n^2$

$$SquareSum \triangleq \lambda p.\ Add\ (Times\ (fst\ \ p)\ (fst\ \ p))\ (Times\ (snd\ \ p)\ (snd\ \ p))$$

This λ-term destructs the pair, squares each and adds them.

If it has a β-normal form then you only have to expand a finite number of times.

(g) Explain why Curry's **Y** combinator is needed and how it works.

There is no "recursion" in λ-calculus since there are no functions.

Curry's **Y** combinator has the effect of passing a function to itself as argument. This allows recursion. Recursion is necessary to implement partial recursive functions; so the usage of Curry's **Y** combinator is an important part in the proof that any computable function is λ-computable.

The IDEA behind fixed point combinators is equivalent to recursion but DON'T use the word recursion to talk about it!

(h) Give a λ-term which is β-equivalent to the $\mathbf{Y}$ combinator, but only uses its f argument once.

$$\mathbf{Y}' \triangleq \lambda\ f.(\lambda\ y.\ y\ y)\ (\lambda\ x.\ f\ (x\ x))$$

5 2009 Paper 6 Question 6

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2009p6q6.pdf

(a) Define what it means for a λ-calculus term to be in normal form. Is it possible for a λ-term to have two normal forms that are not α-equivalent? Provide justification for your answer?

A λ-calculus term is in normal form if there do not exist any β-reductions.

A β-reduction is defined as follows:

$$\frac{}{(\lambda\ x.M)N \to M[N/x]} \quad \frac{M \to M'}{(\lambda x.M)N \to (\lambda\ x.M')N} \quad \frac{N \to N'}{(\lambda x.M)N \to (\lambda\ x.M\ )N'},$$
$$\frac{M \to M'}{M\ N \to M'\ N} \quad \frac{N \to N'}{M\ N \to M\ N'}$$

It's not possible for a λ-term to have two normal forms which are not α-equivalent.

$$\frac{M =_\alpha M'}{M \twoheadrightarrow M'} \quad \frac{M \to M'}{M \twoheadrightarrow M'} \quad \frac{M \twoheadrightarrow M' \quad M' \to M''}{M \twoheadrightarrow M''} \tag{1}$$

Many-step β-reduction is confluent. Formally,

$$M \twoheadrightarrow M' \wedge M \twoheadrightarrow M'' \implies \exists M'''.M' \twoheadrightarrow M''' \wedge M'' \twoheadrightarrow M'''$$

This also applies to many-step β-reduction. So if there was a λ-term which had multiple β normal forms which were not α then confluence for many-step β-reduction *could not* be true. Which is a contradiction. Therefore, the assumption that there exists a λ-term which has multiple β normal forms.

could be more formal

(b) For each of the following, give an example of a λ-term that

(i) is in normal form

$$\lambda\ x.\ x$$

(ii) is not in normal form but has a normal form

$$(\lambda\ x.\ x)y$$

The normal form for this expression is:

$$y$$

I think examiners like closed terms

(iii) does not have a normal form

$$\Omega = (\lambda\ x.\ x\ x)(\lambda\ x.\ x\ x)$$

(c) We define a λ-term N to be *non-trivial* iff there exist A and B such that $N\ A \to^* \mathbf{true}$ and $N\ B \to^* \mathbf{false}$, where $\mathbf{true}$ and $\mathbf{false}$ encode the Booleans.

Give an example of a λ-term that is non-trivial and show that it is non-trivial.

$$N = \lambda\ x.\ x$$

N is *non-trivial* because $N\ \mathbf{true} = \mathbf{true}$ and $N\ \mathbf{false} = \mathbf{false}$.

(d) We define a λ-term N as *total* iff for each λ-term M, either $N\ M \to^* \textbf{true}$ or $N\ M \to^* \textbf{false}$.

Give an example of a λ-term that is total, and show that it is total.

$$\lambda\ x.\textbf{true}$$

Clearly, $\forall M.(\lambda\ x.\textbf{true})M \to \textbf{true}$. So the criteria is fulfilled.

(e) Prove that there is no non-trivial and total λ-term.

Consider a *non-trivial* λ-term N and without loss of generality let A, B be λ-terms such that $N\ A \to^* \textbf{true}$ and $N\ Bto^*\textbf{false}$.

Firstly, use the identity:

$$N(\mathbf{Y}L) =_\beta N(\textbf{if}\ (N\ (\mathbf{Y}\ L))B\ A)$$

I show that $N(\mathbf{Y}\ L)$ can evaluate to neither **true** nor **false** – providing an arbitrary counterexample to the existence of a *non-trivial* and *total* λ-term.

- Case $N(\mathbf{Y}\ L) \to^* \textbf{false}$

$$\begin{aligned} N(\mathbf{Y}\ L) \to^* \textbf{false} &\implies \\ N(\textbf{if}\ (N\ (\mathbf{Y}\ L))B\ A) \to^* \textbf{false} &\implies \\ N(\textbf{if false}\ B\ A) \to^* \textbf{false} &\implies \\ N\ A \to^* \textbf{false} &\implies \\ \textbf{true} \to^* \textbf{false} & \end{aligned}$$

  Since a contradiction has been reached, we can conclude that $N(\mathbf{Y}\ L) \not\to^* \textbf{false}$

- Case $N(\mathbf{Y}\ L) \to^* \textbf{true}$

$$\begin{aligned} N(\mathbf{Y}\ L) \to^* \textbf{true} &\implies \\ N(\textbf{if}\ (N\ (\mathbf{Y}\ L))B\ A) \to^* \textbf{true} &\implies \\ N(\textbf{if true}\ B\ A) \to^* \textbf{true} &\implies \\ N\ B \to^* \textbf{true} &\implies \\ \textbf{false} \to^* \textbf{true} & \end{aligned}$$

  Since a contradiction has been reached, we can conclude that $N(\mathbf{Y}\ L) \not\to^* \textbf{true}$

Therefore, $N(\mathbf{Y}\ L)$ reduces to neither **true** nor **false**. So $N(\mathbf{Y}\ L)$ is not total. So there can exist no λ-term which is both *non-trivial* and total.

(f) What consequences does this have for defining a general equality λ-term such that

equal $A\ B \to^* \textbf{true}$	if $A = B$
equal $A\ B \to^* \cancel{\textbf{true}}$ false	otherwise

This shows that there can exist no general equality λ-term. If there was a general equality λ-term, it would be both *non-trivial* and *total*. By the proof above, there exists no such term and therefore there exists no general equality λ-term.

number of arguments is different!

6 2019 Paper 6 Question 6

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2019p6q6.pdf

(a) (i) Give an inductive definition of the relation $M =_\beta N$ of β-*conversion* between λ-terms M and N.

$$\frac{}{x =_\alpha x} \qquad \frac{Z\#(M\ N)\ M[z/x] =_\alpha N[z/y]}{\lambda x.M =_\alpha \lambda y.N} \qquad \frac{M =_\alpha N \quad M' =_\alpha N'}{MN =_\alpha M'N'}$$

$$\frac{}{(\lambda x.M)N \to M[N/x]} \quad \frac{M \to M'}{\lambda x.M \to \lambda x.M'} \quad \frac{M \to M'}{MN \to M'N}$$
$$\frac{N \to N'}{MN \to MN'} \quad \frac{M =_\alpha N \quad M \to M' \quad M' =_\alpha N}{N \to N'}$$

$$\frac{M =_\alpha N}{M =_\beta N} \quad \frac{M \to N}{M =_\beta N} \quad \frac{M =_\beta M'}{M' =_\beta M} \quad \frac{M =_\beta M' \quad M' =_\beta M''}{M =_\beta M''}$$
$$\frac{M =_\beta M'}{\lambda x.M =_\beta \lambda x.M'} \quad \frac{M =_\beta M' \quad N =_\beta N'}{M\ N =_\beta M'\ N'}$$

(ii) What is meant by a term in β-*normal form*?

A λ-term M is in β-*normal form* if and only if there does not exist any λ-term such that $M \to N$ and $\neg(M =_\alpha N)$.

(iii) If M and N are in β-normal form, explain why $M =_\beta N$ implies that M and N are α-equivalent λ-terms.

Assume $M =_\beta N$ for M, N in β-normal form.

$$\Longrightarrow \exists M'.M \twoheadrightarrow M' \twoheadleftarrow N \qquad \text{by definition}$$

By definition of β-normal form, the only reduction which either can be made is an α-renaming

$$\Longrightarrow \exists M'.M =_\alpha M' =_\alpha N$$
$$\Longrightarrow M =_\alpha N \qquad \text{since } =_\alpha \text{ is an equivalence}$$

(b) Show that there are λ-terms **True**, **False** and **If** satisfying **If True** $M\ N =_\beta M$ and **If False** $M\ N =_\beta N$ and with **True** $\neq$ **False**.

Define the λ-terms as follows:

$$\mathbf{If} \triangleq \lambda fxy.fxy$$
$$\mathbf{True} \triangleq \lambda xy.x$$
$$\mathbf{False} \triangleq \lambda xy.y$$

$$\frac{\dfrac{\mathbf{If\ True}\ M\ N \to \mathbf{True}\ M\ N}{\mathbf{If\ True}\ M\ N =_\beta \mathbf{True}\ M\ N} \quad \dfrac{\mathbf{True}\ M\ N \to M}{\mathbf{True}\ M\ N =_\beta M}}{\mathbf{If\ True}\ M\ N =_\beta M}$$

$$\frac{\dfrac{\mathbf{If\ False}\ M\ N \to \mathbf{False}\ M\ N}{\mathbf{If\ False}\ M\ N =_\beta \mathbf{False}\ M\ N} \quad \dfrac{\mathbf{False}\ M\ N \to N}{\mathbf{False}\ M\ N =_\beta N}}{\mathbf{If\ False}\ M\ N =_\beta N}$$

(c) Define *Curry's fixed point combinator* $\mathbf{Y}$ and prove its fixed-point property.

$$\mathbf{Y} \triangleq \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

The fixed-point property is that for any λ-term M: $\mathbf{Y}M =_\beta M(\mathbf{Y}M)$. A derivation that the fixed-point property holds for any arbitrary λ-term M is given below:

$$\nabla = \frac{\dfrac{\mathbf{Y}M \to (\lambda x.M(x\ x))(\lambda x.M(x\ x))}{\mathbf{Y}M =_\beta (\lambda x.M(x\ x))(\lambda x.M(x\ x))} \quad \dfrac{(\lambda x.M(x\ x))(\lambda x.M(x\ x)) \to M((\lambda x.M(x\ x))(\lambda x.M(x\ x)))}{(\lambda x.M(x\ x))(\lambda x.M(x\ x)) =_\beta M((\lambda x.M(x\ x))(\lambda x.M(x\ x)))}}{\mathbf{Y}M =_\beta M((\lambda x.M(x\ x))(\lambda x.M(x\ x)))}$$

$$\nabla \frac{\dfrac{M(\mathbf{Y}M) \to M((\lambda x.M(x\ x))(\lambda x.M(x\ x)))}{M(\mathbf{Y}M) =_\beta M((\lambda x.M(x\ x))(\lambda x.M(x\ x)))}}{\mathbf{Y}M =_\beta M(\mathbf{Y}M)}$$

(d) Consider the following two properties of λ-term M:

"How does λ-calculus relate to the halting problem?"

(I) there exist λ-terms A and B with $M\ A =_\beta$ **True** and $M\ B =_\alpha$ **False**.

(II) for all λ-terms N, either $M\ N =_\beta$ **True** or $M\ N =_\beta$ **False**.

Prove that M cannot have both properties (I) and (II).

I will assume property (I) holds and perform case analysis to prove that property (II) cannot hold by contradiction, using the identity below:

let A and B be such ....

$$\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A) =_\beta \mathbf{If}\ (M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A)))B\ A$$

Show there exists no nontrivial and total term. Then deduce the halting prolem is unsolvable for λ-calculus since it would have to be total and non-trivial.

- Case $M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A)) =_\beta$ **True**

$$\begin{aligned}\mathbf{True} &=_\beta M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A)) \Longrightarrow\\ \mathbf{True} &=_\beta M(\mathbf{If}\ \mathbf{True}\ B\ A) \Longrightarrow\\ &=_\beta MB \Longrightarrow\\ &=_\beta \mathbf{False} \text{ by definition of } M\end{aligned}$$

L = M(Y(λx.IF(Mx)BA))
= M(IF LBA)

If M L = False then M L = M A = True... Contradiction
If M L = True then M L = M B = False... Contradiction

A contradiction has been reached. Therefore, this case cannot hold!

- Case $M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A)) =_\beta$ **False**

$$\begin{aligned}\mathbf{False} &=_\beta M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A)) \Longrightarrow\\ \mathbf{True} &=_\beta M(\mathbf{If}\ \mathbf{False}\ B\ A) \Longrightarrow\\ &=_\beta MA \Longrightarrow\\ &=_\beta \mathbf{True} \text{ by definition of } M\end{aligned}$$

A contradiction has been reached. Therefore, this case cannot hold!

Therefore $M(\mathbf{Y}(\lambda x.\mathbf{If}\ (M\ x)B\ A))$ is not β-equivalent to either **True** *or* **False**. So if M has property (I) then it cannot have property (II). So there exists no λ-term M which has both property (I) and property (II).

(e) Deduce that there is no λ-term E such that for all λ-terms M and N

$$E\ M\ N =_\beta \begin{cases}\mathbf{True} & \text{if } M =_\beta N\\ \mathbf{False} & \text{otherwise}\end{cases}$$

equal True
nontrivial but total
So cannot exist

It's easier to just use a concrete term.

If such an E existed, it would have both property (I) and property (II). Therefore, no such E can exist using the proof above. take concrete M, e.g. True

Clearly $(EM)\ M =$ **true**, and there exist λ-terms N which are not β-equivalent to M – so $(EM)\ N =$ **false**. So EM is totally defined and takes the value **true** on some inputs and **false** on others. By the proof above, EM cannot exist. M was an arbitrary λ-term and so must exist. We can therefore conclude that E cannot exist.

7 Question 8

(a) Define the λ-term $Fact$ that computes the factorial of a Church numeral.

For this question and the next, I implement recursion by explicitly passing the λ-term a reference to itself using Curry's $\mathbf{Y}$ combinator.

$$Fact \triangleq \mathbf{Y}(\lambda\ f\ n.if\ (Eq_0\ \ n)\ 1\ (Times\ \ n\ (\mathbf{Y}\ f\ (Pred\ \ n))))$$

let's discuss

(b) Define the λ-term Fib such that $Fib\, n =_\beta Fib_n$ where F_n is the n-th Fibonacci number defined recursively as $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$.

$$Fib \triangleq \mathbf{Y}(\lambda\, f\, n.\ if\ (Eq_0\ n)\, 0\, (if\ (Eq_0\ (Pred\ n))\, 1\, (Add\, (\mathbf{Y}\, f\ (Pred\ n))(\mathbf{Y}\, f\ (Pred\, (Pred\ n))))))$$

not needed!
f = YM already!

Use the Y combinator to give functions a reference to themselves

Y (λfx. IF x < = 1 then 1 else f (x - 1))...

β-nf
a λ-term which cannot be reduced to anything which is not α-equivalent to itself

Church Rosser Theorem
many-step β-reduction is confluent
not GUARANTEED to converge (if no β-normal form) but always POSSIBLE

Normal order reduction
A way to do reductions in some order which guarantees we reach β-normal form if it exists
Otherwise, we are not guaranteed to reach β-normal form

λ-definable
If the function is undefined then the application has no β-normal form
If the function is defined then the application term is β-equivalent to the output of the function

Minimisation is represented by using a fixed point combinator (i.e. Curry's Y combinator)
Y M =_β M (Y M)
So we can do recursion! i.e. minimisation

normal order reduction
leftmost - outermost reduction
guaranteed to find a β-normal form if it exists

λ-calculus is lazy!
we don't have to evaluate arguments before a substitution!

For all equivalences
If M =_δ M' then in all contexts C[M] =_
This is just a fact of equivalences... maybe don't assume it in an exam but bear it in mind

Notice similarities between church numerals and primitive recursion!
\underline{m} is primitive recursion m times where f is the inductive case and x is the base case!
Note the other parameters are passed by name (power of λ-terms)

In partial recursive functions
we always evaluate the arguments without evaluating them
In λ-calculus we can do substituation without evaluating them

Intuition: we encode booleans as conditionals but with the predicate filled

If is η-equivalent to the identity function
It just takes the arguments and applies them in the same order

Or = λab. IF a True b
And = λab. IF a b False

Memorise the definition for Pair
PAIR = λxyp. p x y
(x, y) = λp. p x y
First = True
Second = False