Algunas consideraciones y ejercicios sobre $\lambda-$cálculo

NOTA PREVIA. Como el **if** no es más que la función identidad en tres variables, puede obviarse en todos los casos. Por eso, en lo que sigue no habrá ninguno.

Ejercicio 1. Definir un $\lambda-$término **implica** que codifique el conectivo lógico $\longrightarrow$. Comprobar la corrección de la codificación.

La implicación, si el antecedente es falso es siempre verdadera. En cambio, si el antecedente es cierto el valor de la implicación es el de la consecuencia. Con esto en mente y teniendo en cuenta que las definiciones de **true** y **false**, definimos

$$\textbf{implica} \equiv \lambda pq.p\ q\ \textbf{true}$$

No vamos a probar que la codificación es correcta en su totalidad. Para dar una idea, probaremos únicamente que **implica true false** $\twoheadrightarrow$ **false**. Lo haremos especificando todas y cada una de las reducciones realizadas.

$$\textbf{implica true false} \equiv (\lambda pq.p\ q\ \textbf{true})\ \textbf{true false} \equiv$$

$$\equiv ((\lambda p.(\lambda q.p\ q\ \textbf{true}))\ \textbf{true})\ \textbf{false} \rightarrow (\lambda q.\textbf{true}\ q\ \textbf{true})\ \textbf{false} \rightarrow$$

$$\rightarrow \textbf{true false true} \equiv (\lambda xy.x)\ \textbf{false true} \equiv ((\lambda x.(\lambda y.x))\ \textbf{false})\ \textbf{true} \rightarrow$$

$$\rightarrow (\lambda y.\textbf{false})\ \textbf{true} \rightarrow \textbf{false}.$$

Vamos a ver un par de ejemplos sobre la simulación de la recursividad en el $\lambda-$cálculo, pero antes, a modo de recordatorio y porque lo vamos a utilizar después, haremos un ejercicio previo.

Ejercicio 2. Codificar, en $\lambda-$cálculo, la lista $L = [1]$. Comprobar que no es vacía, que su cola es la lista vacía y que su cabeza es el número 1 usando los $\lambda-$términos definidos en los apuntes (página 60).

Empezamos la codificación de la lista dada que denotaremos por **l**. Con esta notación,

$$\textbf{l} \equiv \textbf{cons}\ \underline{1}\ \textbf{nil} \equiv \textbf{pair false}\ (\textbf{pair}\ \underline{1}\ \textbf{nil}) \equiv (\lambda xyf.fxy)\ \textbf{false}\ (\textbf{pair}\ \underline{1}\ \textbf{nil}) \twoheadrightarrow$$

$$\twoheadrightarrow \lambda f.f\ \textbf{false}\ (\textbf{pair}\ \underline{1}\ \textbf{nil}) \equiv \lambda f.f\ \textbf{false}\ ((\lambda xyg.gxy)\ \underline{1}\ \textbf{nil}) \twoheadrightarrow \lambda f.f\ \textbf{false}\ (\lambda g.g\ \underline{1}\ \textbf{nil})$$

Una vez codificada la lista ($\mathbf{l} \equiv \lambda f.f\ \mathbf{false}\ (\lambda g.g\ \underline{1}\ \mathbf{nil})$), comprobamos que no es vacía:

$$\mathbf{null\ l} \equiv \mathbf{fst\ l} \equiv (\lambda p.p\ \mathbf{true})\ \mathbf{l} \rightarrow\ \mathbf{l\ true} \equiv (\lambda f.f\ \mathbf{false}\ (\lambda g.g\ \underline{1}\ \mathbf{nil}))\ \mathbf{true} \rightarrow$$

$$\rightarrow \mathbf{true\ false}\ (\lambda g.g\ \underline{1}\ \mathbf{nil}) \twoheadrightarrow \mathbf{false}$$

Calculemos la cola de $\mathbf{l}$:

$$\mathbf{tl\ l} \equiv (\lambda z.\mathbf{snd}(\mathbf{snd}\ z))\ \mathbf{l} \rightarrow \mathbf{snd}(\mathbf{snd\ l}) \equiv \mathbf{snd}((\lambda p.p\ \mathbf{false})\ \mathbf{l}) \rightarrow \mathbf{snd}(\mathbf{l\ false}) \equiv$$

$$\equiv \mathbf{snd}((\lambda f.f\ \mathbf{false}\ (\lambda g.g\ \underline{1}\ \mathbf{nil}))\ \mathbf{false}) \rightarrow \mathbf{snd}(\mathbf{false\ false}\ (\lambda g.g\ \underline{1}\ \mathbf{nil})) \twoheadrightarrow$$

$$\twoheadrightarrow \mathbf{snd}(\lambda g.g\ \underline{1}\ \mathbf{nil}) \equiv (\lambda p.p\ \mathbf{false})(\lambda g.g\ \underline{1}\ \mathbf{nil}) \rightarrow (\lambda g.g\ \underline{1}\ \mathbf{nil})\ \mathbf{false} \rightarrow$$

$$\rightarrow \mathbf{false}\ \underline{1}\ \mathbf{nil} \twoheadrightarrow \mathbf{nil}$$

Finalmente, computamos la cabeza de $\mathbf{l}$:

$$\mathbf{hd\ l} \equiv (\lambda z.\mathbf{fst}(\mathbf{snd}\ z))\ \mathbf{l} \rightarrow \mathbf{fst}(\mathbf{snd\ l}) \twoheadrightarrow {}^{1}\mathbf{fst}(\lambda g.g\ \underline{1}\ \mathbf{nil}) \equiv$$

$$\equiv (\lambda p.p\ \mathbf{true})(\lambda g.g\ \underline{1}\ \mathbf{nil}) \rightarrow (\lambda g.g\ \underline{1}\ \mathbf{nil})\ \mathbf{true} \rightarrow \mathbf{true}\ \underline{1}\ \mathbf{nil} \twoheadrightarrow \underline{1}$$

Como no se puede utilizar un $\lambda-$término para definirse a sí mismo, para simular la recursividad en el $\lambda-$cálculo se utilizan los combinadores de punto fijo. En los dos ejemplos que haremos a continuación se utiliza el combinador paradójico, $\mathbf{Y}$, pero podría utilizarse cualquier otro. Empezamos con el factorial, que está definido en los apuntes.

Ejercicio 3. Utilizar el $\lambda-$término **fact** definido en los apuntes para calcular el factorial de 2. (**NOTA**: En lo que sigue hay varias reducciones que convendría que se hiciesen en detalle como "entrenamiento". Por ejemplo, $\mathbf{pre}\ \underline{2} \twoheadrightarrow \underline{1}$ o $\mathbf{iszero}\ \underline{2} \twoheadrightarrow \mathbf{false}$. En todo caso, como el principal interés está en la recursividad, me las saltaré).

Sea $\mathbf{F} \equiv \lambda gn.(\mathbf{iszero}\ n)\ \underline{1}\ (\mathbf{mult}\ n\ (g(\mathbf{pre}\ n)))$[2]. Por lo tanto, $\mathbf{fact} \equiv \mathbf{YF}$. En consecuencia,

$$\mathbf{fact}\ \underline{2} \equiv \mathbf{YF}\ \underline{2} = {}^{3}\mathbf{F}(\mathbf{YF})\ \underline{2} \equiv \mathbf{F fact}\ \underline{2} \twoheadrightarrow$$

$$\twoheadrightarrow (\mathbf{iszero}\ \underline{2})\ \underline{1}\ (\mathbf{mult}\ \underline{2}\ (\mathbf{fact}\ (\mathbf{pre}\ \underline{2}))) \twoheadrightarrow \mathbf{false}\ \underline{1}\ (\mathbf{mult}\ \underline{2}\ (\mathbf{fact}\ (\mathbf{pre}\ \underline{2}))) \twoheadrightarrow$$

$$\twoheadrightarrow \mathbf{mult}\ \underline{2}\ (\mathbf{fact}\ (\mathbf{pre}\ \underline{2})) \twoheadrightarrow \mathbf{mult}\ \underline{2}\ (\mathbf{fact}\ \ \underline{1}) \equiv \mathbf{mult}\ \underline{2}\ (\mathbf{YF}\ \underline{1}) =$$

[1] La reducción $\mathbf{snd\ l} \twoheadrightarrow \lambda g.g\ \underline{1}\ \mathbf{nil}$ se ha hecho antes.

[2] Obsérvese que, como se había anunciado, hemos obviado el **if**.

[3] Aquí no cabe reducción alguna pues la igualdad es resultado de reducciones y expansiones. Si se utilizase el combinador de punto de fijo Θ de Turing, por ejemplo, podríamos sustituir la igualdad por $\twoheadrightarrow$.

$$= \textbf{mult } \underline{2} \ (\textbf{F}(\textbf{YF}) \ \underline{1}) \equiv \textbf{mult } \underline{2} \ (\textbf{F fact } \underline{1}) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{mult } \underline{2} \ ((\textbf{iszero } \underline{1}) \ \underline{1} \ (\textbf{mult } \underline{1} \ (\textbf{fact pre } \underline{1}))) \twoheadrightarrow \textbf{mult } \underline{2} \ (\textbf{false } \underline{1} \ (\textbf{mult } \underline{1} \ (\textbf{fact pre } \underline{1}))) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ (\textbf{fact pre } \underline{1})) \twoheadrightarrow \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ (\textbf{fact } \underline{0})) \equiv \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ (\textbf{YF } \underline{0})) =$$

$$= \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ (\textbf{F}(\textbf{YF}) \ \underline{0})) \equiv \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ (\textbf{F fact } \underline{0})) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ ((\textbf{iszero } \underline{0}) \ \underline{1} \ (\textbf{mult } \underline{0} \ (\textbf{fact } (\textbf{pre } \underline{0}))))) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{mult } \underline{2} \ (\textbf{mult } \underline{1} \ \underline{1}) \twoheadrightarrow \textbf{mult } \underline{2} \ \underline{1} \twoheadrightarrow \underline{2}$$

Ejercicio 4. Definir un $\lambda-$término **long** que aplicado a una lista codificada en $\lambda-$cálculo, calcule la longitud de la misma. Utilizarla para calcular la longitud de la lista $L = [1]$. (Este ejercicio está extraído del examen de 21 de mayo de 2018).

La definición recursiva de una lista es como sigue: cero si la lista es vacía o, si no, uno más la longitud de su cola. Siguiendo esta definición sea $\text{G} \equiv \lambda gl.(\textbf{null } l) \ \underline{0} \ (\textbf{suc } (g(\textbf{tl } l)))$. Ahora, $\textbf{long} \equiv \textbf{YG}$. Con esta definición, calculemos la longitud de la lista que hemos codificado en el Ejercicio 1, esto es, $\textbf{l} \equiv \lambda f.f \ \textbf{false } (\lambda g.g \ \underline{1} \ \textbf{nil})$.[4]

$$\textbf{long l} \equiv \textbf{YG l} = \textbf{G}(\textbf{YG}) \ \textbf{l} \equiv \textbf{G long l} \twoheadrightarrow (\textbf{null l}) \ \underline{0} \ (\textbf{suc } (\textbf{long}(\textbf{tl l}))) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{false } \underline{0} \ (\textbf{suc } (\textbf{long}(\textbf{tl l}))) \twoheadrightarrow \textbf{suc } (\textbf{long } (\textbf{tl l}))) \twoheadrightarrow \textbf{suc } (\textbf{long nil}) \equiv$$

$$\equiv \textbf{suc } (\textbf{YG nil}) = \textbf{suc } (\textbf{G}(\textbf{YG}) \ \textbf{nil}) \equiv \textbf{suc } (\textbf{G long nil}) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{suc } (\textbf{null nil}) \ \underline{0} \ (\textbf{suc } (\textbf{long}(\textbf{tl nil}))) \twoheadrightarrow \textbf{suc } (\textbf{true } \underline{0} \ (\textbf{suc } (\textbf{long}(\textbf{tl nil}))) \twoheadrightarrow$$

$$\twoheadrightarrow \textbf{suc } \underline{0} \twoheadrightarrow \underline{1}$$

[4] Haremos uso de los cálculos efectuados en el Ejercicio 1 y también de la reducción $\textbf{suc } \underline{0} \twoheadrightarrow \underline{1}$.