Lenguajes de Programación, 2019-1
Nota adicional: Representación anónima de términos λ (índices de de-Bruijn)

Favio E. Miranda Perea   Lourdes del Carmen González Huesca
Facultad de Ciencias UNAM

23 de noviembre de 2018

1. Idea

Dada una abstracción lambda, la idea es representar una variable ligada apuntando directamente al símbolo lambda que la liga en el árbol de sintaxis abstracta correspondiente, es decir mediante el número de lambdas que es necesario "saltar" hasta encontrar la lambda que liga a la variable en cuestión; dichos números son conocidos como *índices de de-Bruijn*[1]. Veamos unos ejemplos:

$$\begin{array}{rll} \lambda x.x & \text{se representa con} & \lambda.0 \\ \lambda s\lambda z.z & \text{se representa con} & \lambda.\lambda.0 \\ \lambda s\lambda z.sz & \text{se representa con} & \lambda.\lambda.10 \\ \lambda m\lambda n\lambda s\lambda z.ms(nsz) & \text{se representa con} & \lambda.\lambda.\lambda.\lambda.31(210) \end{array}$$

Como podemos ver, la representación anónima de un término cerrado es única, lo cual no sucede si tenemos variables libres. Por ejemplo en el término $\lambda z.zxy$ lo único que podemos obtener con el proceso recien descrito es $\lambda.0xy$. Para poder representar anónimamente términos con variables libres es necesario declarar qué índices representarán tales variables mediante un contexto de índices. Por ejemplo si definimos $x \mapsto 4, y \mapsto 5, z \mapsto 6$ entonces $\lambda z.zxy$ se convierte en $\lambda.045$, $y(xz)$ es $5(46)$ y $\lambda s\lambda z.sy(zx)$ se representa con $\lambda.\lambda.15(04)$.
Por supuesto que en el caso de que el término no esté cerrado, su representación no es única, pero se puede llegar a una convención para asignar índices a las variables libres de una manera determinista:

> dado un término t asignar a cada presencia de una variable libre, empezando desde la **derecha**, el menor índice mayor o igual al mínimo número de saltos (lambdas) necesarios para liberar a la variable de todos los alcances dentro de los que se encuentre y que ese índice no haya sido utilizado hasta ese momento para nombrar variables libres.

Ejemplo 1.1

$$\begin{array}{rll} \lambda z.zxy & \textit{se representa con} & \lambda.021 \\ \lambda z.zx(\lambda y.zxy) & \textit{se representa con} & \lambda.01(\lambda.120) \end{array}$$

[1]Nicolaas Govert de Bruijn (1918-2012), matemático holandes, la pronunciación mas cercana al holandes es "de Brown"

En el primer caso para la y basta con saltar la única lambda visible, de manera que la presencia de y se representa con 1 y la de x con 2 pues el 1 ya fue utilizado para y.
En el segundo caso la segunda presencia de x se representa con 2 pues estaba bajo el alcance de las dos lambdas presentes, es decir, se necesita saltar dos lambdas para liberarla. Por otro lado la primera presencia de x se representa con 1 dado que se necesita sólo un salto para liberarla y el 1 no ha sido usado para variables libres.

En nuestra experiencia el proceso para asignar índices a variables libres es más simple si se trabaja directamente en los árboles de sintaxis abstracta como sigue:

1. Sea $x_n, x_{n-1}, \ldots, x_0$ la sucesión sin repeticiones de las variables libres del término t, donde si habia repeticiones de x_i, entonces se mantiene únicamente la última presencia de dicha variable en la sucesión. Por ejemplo, si la sucesión de variables libres es x, y, z, x, v, y, u, entonces la sucesión sin repeticiones es z, x, v, y, u.

2. El término $\lambda x_n.\lambda x_{n-1}.\ldots.\lambda x_0.t$ es cerrado por lo que los índices se pueden asignar sin apelar a un contexto, obteniendose el término anónimo $\lambda.\ldots.\lambda.a_t$.

3. En tal caso al eliminar las $n+1$ abstracciones agregadas en el paso anterior (a las cuales nos referimos como abstracciones fantasma), obtenemos el término a_t que es la representación anónima de t.

Ejemplo 1.2 *La representación anónima para los términos del ejemplo anterior se obtiene como sigue:*

1. $t =_{def} \lambda z.zxy$ *tiene como sucesión de variables libres a* x, y.

   a) *El término* $\lambda x.\lambda y.\lambda z.zxy$ *se representa de manera única con* $\lambda.\lambda.\lambda.021$

   b) *Por lo tanto* t *se representa con* $\lambda.021$.

2. $t =_{def} \lambda z.zx(\lambda y.zxy)$ *tiene como sucesión de variables libres a* x.

   a) *El término* $\lambda x.\lambda z.zx(\lambda y.zxy)$ *se representa de manera única con* $\lambda.\lambda.01(\lambda.120)$

   b) *Por lo tanto* t *se representa con* $\lambda.01(\lambda.120)$

Es de suma importancia recalcar que la asignación de índices de de-Bruijn NO es una sustitución textual, de hecho una misma variable, ya sea libre o ligada, puede recibir distintos índices como el último ejemplo lo muestra, las dos presencias de la variable ligada z tienen índices 0 y 1, mientras que las presencias de la variable libre x aparecen con índices 1 y 2 respectivamente.

Ciertamente la representación anónima no es amigable para un usuario, su aplicación principal es en la implementación del cálculo lambda y por lo tanto en la implementación de constructores en cualquier lenguaje, los cuales requieran de un ligado, como son el operador `case` para tipos suma o el fix para definiciones recursivas.

Pasemos ahora a las formalidades.

2. Los índices de de-Bruijn

Los términos anónimos se definen como sigue:

$$a ::= n \mid aa \mid \lambda.a$$

Definición 1 (Contexto de índices) *Dado un conjunto de variables* $V = \{x_1, \ldots, x_n\}$ *un contexto de índices para* V *es una asignación de índices de de-Bruijn* $\Gamma := x_1 \mapsto m_1, \ldots, x_n \mapsto m_n$ *a las variables de* V. *Dado un contexto* Γ, *denotamos con* $\mathsf{dom}\,\Gamma$ *al conjunto* V *de nombres de variables.*

Definición 2 (Contextos canónicos) *Dado un término* t *definimos su contexto canónico como sigue: sea* $x_n, \ldots, x_0$ *la sucesión de variables libres de* t *tomadas en el mismo orden que figuran en* t, *donde en caso de haber repeticiones se mantiene únicamente la última presencia de una variable libre. Entonces el contexto canónico esta dado por la asignación* $x_n \mapsto n$.
Puesto que la asignación está fija por el orden en que figuran las variables en t, *el contexto canónico se denota simplemente como* $\Gamma = x_n, \ldots, x_0$.
Dado un contexto canónico $\Gamma = x_n, \ldots, x_0$ *y una variable* x, *no necesariamente distinta a las* x_i, *denotamos con* Γ, x *al contexto* $\Gamma = x_n, \ldots, x_0, x$ *es decir al contexto canónico* $x_n \mapsto n+1, x_{n-1} \mapsto n, \ldots, x_0 \mapsto 1, x \mapsto 0$

Ejemplo 2.1 *Veamos los contextos canónicos para algunos términos:*

- *Para el término* $\lambda z.zxy$ *tenemos el contexto canónico* $\Gamma = x, y$, *es decir* $x \mapsto 1, y \mapsto 0$.
- *Para el término* $\lambda z.zx(\lambda y.zxy)$ *el contexto canónico es* $\Gamma = x$, *es decir* $x \mapsto 0$.
- *Para el término* $\lambda x \lambda y.uxyzzyv$ *el contexto canónico es* $\Gamma = u, z, v$, *es decir* $u \mapsto 2, z \mapsto 1, v \mapsto 0$.
- *Para el término* $xyzx$ *el contexto canónico es:* $\Gamma = y, z, x$, *es decir* $y \mapsto 2, z \mapsto 1, x \mapsto 0$.
- *Para el término* $uvxyzxxv$ *el contexto canónico es* $\Gamma = u, y, z, x, v$, *es decir* $u \mapsto 4, y \mapsto 3, z \mapsto 2, x \mapsto 1, v \mapsto 0$.

En adelante trabajaremos únicamente con contextos canónicos.

3. Eliminación y restauración de variables

3.1. La función qn

Definición 3 *La función "quita nombres" que toma un contexto de índices* Γ *y un término* t *del cálculo lambda y nos devuelve su representación anónima se define como sigue:*

$$\begin{array}{lcl}
\mathsf{qn}(\Gamma; x) & = & \textit{el índice de la presencia más a la derecha de } x \textit{ en } \Gamma \\
\mathsf{qn}(\Gamma; \lambda x.t) & = & \lambda.\,\mathsf{qn}(\Gamma, x\,; t) \\
\mathsf{qn}(\Gamma; rs) & = & \mathsf{qn}(\Gamma; r)\,\mathsf{qn}(\Gamma; s)
\end{array}$$

Sea $t := \lambda z.zx(\lambda y.zxy)$ entonces tenemos $\Gamma = x$ y

$$\begin{array}{rcl}
\mathsf{qn}(\Gamma;t) & = & \lambda.\,\mathsf{qn}\big(\Gamma,z\,;zx(\lambda y.zxy)\big) \\
& = & \lambda.\,\mathsf{qn}(\Gamma,z\,;zx)\,\mathsf{qn}(\Gamma,z\,;\lambda y.zxy) \\
& = & \lambda.\,\mathsf{qn}(\Gamma,z\,;z)\,\mathsf{qn}(\Gamma,z\,;x)\,\mathsf{qn}(\Gamma,z\,;\lambda y.zxy) \\
& = & \lambda.01\,\mathsf{qn}(\Gamma,z\,;\lambda y.zxy) \\
& = & \lambda.01\big(\lambda.\,\mathsf{qn}(\Gamma,z,y;zxy)\big) \\
& = & \lambda.01\big(\lambda.\,\mathsf{qn}(\Gamma,z,y;zx)\,\mathsf{qn}(\Gamma,z,y\,;y)\big) \\
& = & \lambda.01\big(\lambda.\,\mathsf{qn}(\Gamma,z,y;zx)0\big) \\
& = & \lambda.01\big(\lambda.\,\mathsf{qn}(\Gamma,z,y;z)\,\mathsf{qn}(\Gamma,z,y\,;x)0\big) \\
& = & \lambda.01\big(\lambda.120\big)
\end{array}$$

3.2. La función pn

Para definir la función inversa de qn, la cual dado un término anónimo devuelve un término usual del cálculo lambda, necesitamos de contextos de nombres, es decir contextos de la forma $\Delta = \{n \mapsto x_n, \dots, 0 \mapsto x_0\}$ denotados simplemente como $\Delta = x_n, \dots, x_0$ sobreentendiendo que la variable más a la derecha es el nombre para el índice 0 y así sucesivamente[2] . Además se requiere de la siguiente convención: suponemos que el conjunto de variables está ordenado de manera que siempre tendrá sentido la frase: *la primera variable que no figura en el dominio de* Δ; esto con ánimos de mantener el determinismo aunque, desde el punto de vista puramente matemático nos bastaría con pedir una variable nueva.

Definición 4 *La función "pon nombres" que toma un contexto de nombres Δ y un término anónimo a y nos devuelve un término correspondiente a a en el cálculo lambda se define como sigue:*

$$\begin{array}{rcl}
\mathsf{pn}(\Delta;k) & = & \textit{el nombre correspondiente a } k \textit{ en } \Delta \\
\mathsf{pn}(\Delta;\lambda.a) & = & \lambda x.\,\mathsf{pn}(\Delta,x;a) \quad \textit{siendo } x \textit{ el primer nombre que no figura en } \Delta \\
\mathsf{pn}(\Delta;ab) & = & \mathsf{pn}(\Delta;a)\,\mathsf{pn}(\Delta;b)
\end{array}$$

Claramente la función pn es parcial y solo devolverá un término del cálculo lambda dado un contexto adecuado, en particular dado un término anónimo a, un contexto adecuado necesita tener al menos el número de variables libres de a.
Una variable libre se puede reconocer verificando el número de lambdas a la izquierda de un índice dado. Aunque en nuestra opinión la identificación de índices correspondientes a variables libres en un término anónimo es más simple siguiendo este procedimiento:

1. Dado el término anónimo a, construir su árbol de sintaxis abstracta T. Las hojas de T estarán etiquetadas con índices.

2. Agregar el apuntador correspondiente a cada hoja de T de acuerdo al índice de dicha hoja, empezando desde la hoja más a la izquierda. Será necesario añadir tantas abstracciones (lambdas) fantasma a la raiz de T como sea necesario para que los apuntadores estén bien definidos.

[2] Obsérvese que la notación para contextos de nombres Δ es la misma que para contextos de índices Γ, a saber una sucesión de variables $x_n, \dots, x_0$. Sin embargo en el primer caso los nombres de variables los proponemos nosotros, mientras que en el segundo los nombres se obtienen del término en cuestión.

3. El número de abstracciones (lambdas) fantasma agregadas a la raiz de T es el número de variables libres en a.

Ejemplo 3.1 *Sean* $\Delta = x$ *y* $a =_{def} \lambda.01(\lambda.120)$*. Entonces:*

$$
\begin{aligned}
\mathsf{pn}(\Delta\,; a) &= \lambda z.\, \mathsf{pn}\left(\Delta, z\,; 01(\lambda.120)\right) \\
&= \lambda z.\, \mathsf{pn}(\Delta, z\,; 01)\, \mathsf{pn}(\Delta, z\,; \lambda.120) \\
&= \lambda z.\, \mathsf{pn}(\Delta, z\,; 0)\, \mathsf{pn}(\Delta, z\,; 1)\, \mathsf{pn}(\Delta, z\,; \lambda.120) \\
&= \lambda z.zx\, \mathsf{pn}(\Delta, z\,; \lambda.120) \\
&= \lambda z.zx\big(\lambda w.\, \mathsf{pn}(\Delta, z, w\,; 120)\big) \\
&= \lambda z.zx\big(\lambda w.\, \mathsf{pn}(\Delta, z, w\,; 1)\, \mathsf{pn}(\Delta, z, w\,; 2)\, \mathsf{pn}(\Delta, z, w\,; 0)\big) \\
&= \lambda z.zx\big(\lambda w.zxw\big)
\end{aligned}
$$

En adelante escribimos $t \approx_{dB} a$ para denotar que a es la representación anónima de t obtenida al usar el contexto canónico.

3.3. Relación entre pn y qn

Intuitivamente las funciones que eliminan y restauran nombres deben ser inversas. La relación exacta es la siguiente.

Proposición 1 *Si* t *es un* λ*-término con contexto canónico de variables libres* Γ *entonces*

$$\mathsf{pn}(\Gamma; \mathsf{qn}(\Gamma; t)) \equiv_\alpha t$$

Demostración. Inducción sobre t.

Obsérvese que se debe utilizar exactamente el mismo contexto para quitar y poner nombres. De otra manera se cambiarían los nombres de las variables libres lo cual es inaceptable. En una implementación se guardará el contexto Γ obtenido del término original t de manera que esté disponible para restaurar posteriormente los nombres de variables libres.

Proposición 2 *Si* a *es un término anónimo y* Δ *es un contexto de nombres para sus variables libres entonces*

$$\mathsf{qn}(\Delta; \mathsf{pn}(\Delta; a)) = a$$

Demostración. Inducción sobre a.

4. Sustitución

El objetivo de esta sección es definir la sustitución $a[n := b]$, se sustituye a la variable con índice n por el término b en a. Es necesario ser muy precavido con los índices de las variables libres.
Cuando una sustitución opera en el cuerpo de una abstracción como $(\lambda.2)[1 := s]$ el contexto interior tiene una variable más que la original (la variable que estaba ligada por la abstracción), de forma que es necesario incrementar los índices de las variables libres en s para que se sigan refiriendo a las

mismas variables que antes de la sustitución. Por ejemplo si el término $2(\lambda.0)$ en el contexto $z \mapsto 2$, es decir $z(\lambda w.w)$ se va a sustituir dentro de una abstracción es necesario incrementar el índice 2 a 3 pero no el índice 0.
Considerese la reducción

$$\big(\lambda x.(\lambda y.\underline{x}\,y)z\underline{x}\big)(\lambda w.vw) \to_\beta \big(\lambda y.\underline{(\lambda w.vw)}\,y\big)z\underline{(\lambda w.vw)}$$

El redex se representa con $\big(\lambda.(\lambda.\underline{1}\,0)2\underline{0}\big)(\lambda.10)$ mientras que el reducto debe representarse con

$$\big(\lambda.\underline{(\lambda.20)}\,0\big)1\underline{(\lambda.10)}$$

Obsérvese que:

- Las variables libres no afectadas por la sustitución (la z en el ejemplo, cuyo índice es 2) ven su índice decrementado (el primer 1 en el reducto), reflejando el hecho de que una lambda desaparece.
- Los índices de las variables libres del término a sustituir (en este caso el 1 en $\lambda.10$) deben ajustarse en cada posición (en este caso las posiciones de x con los índices subrayados 1 y 0 en el redex), de acuerdo a si quedan bajo más o menos lambdas.

El proceso formal de sustitución se hace con la ayuda de una función auxiliar que desplaza los índices necesarios, dicha función toma un parametro de corte c que controla qué variables deben incrementarse. Dicho parámetro inicia en 0, lo que significa que todas las variables deben ajustarse, y cada vez que el proceso de ajuste se realiza bajo una abstracción el parámetro de corte se incrementa en uno.

Definición 5 *La función* shift *se define como sigue:*

$$\begin{array}{lcl} \mathsf{shift}(d,c,k) & = & \mathsf{if}\ k < c\ \mathsf{then}\ k\ \mathsf{else}\ k+d \\ \mathsf{shift}(d,c,\lambda.t) & = & \lambda.\,\mathsf{shift}(d,c+1,t) \\ \mathsf{shift}(d,c,rs) & = & \mathsf{shift}(d,c,r)\,\mathsf{shift}(d,c,s) \end{array}$$

Con ayuda de la función shift ya podemos definir la sustitución en términos anónimos.

Definición 6 *La sustitución en términos anónimos $a[j := s]$ se define como sigue:*

$$\begin{array}{rcl} n[j := s] & = & \mathsf{if}\ n = j\ \mathsf{then}\ s\ \mathsf{else}\ n \\ (\lambda.t)[j := s] & = & \lambda.t[j+1 := \mathsf{shift}(1,0,s)] \\ (tr)[j := s] & = & t[j := s]r[j := s] \end{array}$$

Esta definición depende de un contexto dado Γ que sea fijo para poder determinar los índices de las variables libres en el término anónimo a.

Veamos un ejemplo, se tiene que

$$(\lambda y.yzx)[z := (\lambda x.xz)] = \lambda y.y(\lambda x.xz)x \tag{1}$$

El contexto canónico es $\Gamma = z, x$ es decir, $z \mapsto 1, x \mapsto 0$ y tenemos que:

$$\begin{array}{rcl}
\lambda y.yzx & \approx_{dB} & \lambda.021 \\
\lambda x.xz & \approx_{dB} & \lambda.02 \\
\lambda y.y(\lambda x.xz)x & \approx_{dB} & \lambda.0(\lambda.03)1
\end{array}$$

Es importante observar que en el proceso de transformación de más de un término a términos anónimos debe usarse un mismo contexto para indizar las variables libres de todos los términos en cuestión, en otro caso la representación será erronea. Por ejemplo en el segundo término arriba, bastaría con el contexto $\Gamma = z$ pero los otros dos términos requieren de $\Gamma = z, x$. Por eso la representación de $\lambda x.xz$ es $\lambda.02$ y NO $\lambda.01$.

La sustitución correspondiente a (**??**) en términos anónimos es

$$(\lambda.021)[1 := \lambda.02] = \lambda.0(\lambda.03)1 \tag{2}$$

Obsérvese que es **incorrecto** pensar que se debe sustituir el número correspondiente a la variable en la abstracción lambda, es decir, realizar la sustitución $(\lambda.021)[2 := \lambda.02]$. Lo correcto es usar el contexto canónico para obtener el índice correspondiente a la variable que se va a sustituir.
Como se espera, la sustitución anónima no permite sustituir variables ligadas ya que por ejemplo, si se intenta $(\lambda x.xy)[x := t]$ al eliminar nombres obtenemos $(\lambda.01)$ pero la sustitución $[x := t]$ no está definida pues no hay un índice para x en el contexto canónico que en este caso es $\Gamma = y$.

Ejemplo 4.1 *Vamos a verificar que la definición de sustitución anónima funciona correctamente para el ejemplo de arriba :*

$$\begin{array}{rcl}
(\lambda.021)[1 := \lambda.02] & = & \lambda.(021)\big[2 := \mathsf{shift}(1,0,\lambda.02)\big] \\
& = & \lambda.(021)\big[2 := \lambda.\,\mathsf{shift}(1,0+1,02)\big] \\
& = & \lambda.(021)\big[2 := \lambda.\,\mathsf{shift}(1,1,0)\,\mathsf{shift}(1,1,2)\big] \\
& = & \lambda.(021)[2 := \lambda.03] \\
& = & \lambda.\big(0[2 := \lambda.03]2[2 := \lambda.03]1[2 := \lambda.03]\big) \\
& = & \lambda.0(\lambda.03)1 \\
& \approx_{dB} & \lambda y.y(\lambda x.xz)x
\end{array}$$

5. β-reducción

Para poder implementar la semántica operacional del cálculo lambda tenemos que definir la regla de β-reducción en términos anónimos. El lector debe verificar que la siguiente definición funciona correctamente:

$$(\lambda.t)s \to_\beta \mathsf{shift}\big(-1, 0, t[0 := \mathsf{shift}(1,0,s)]\big)$$

La idea detrás de esta definición es que en la reducción $(\lambda x.t)s \to_\beta t[x := s]$ la variable ligada x en el redex desaparece en el reducto de manera que hay que reindizar las variables en el término $t[x := s]$ tomando en cuenta que la variable x ya no pertenece al contexto. Esto corresponde a disminuir en 1 las variables libres en el reducto, de ahí el -1 en la llamada a la función shift.
De forma similar, las variables libres en s deben ser incrementadas en 1 antes de hacer la sustitución en t dado que t requiere de un contexto más grande para estar bien definido, puesto que en t se

necesita un índice para x. Por ejemplo $(\lambda.102)(\lambda.0)$ debe reducirse a $0(\lambda.0)1$ y NO a $1(\lambda.0)2$.

Veamos como ejemplo la reducción

$$\big(\lambda z.(\lambda u.z)z\big)(\lambda w.xyw) \to_\beta (\lambda u.(\lambda w.xyw))(\lambda w.xyw).$$

Los términos anónimos correspondientes son:

$$\begin{array}{rcl} \big(\lambda z.(\lambda u.z)z\big)(\lambda w.xyw) & \approx_{dB} & \big(\lambda.(\lambda.1)0\big)(\lambda.210) \\ (\lambda u.(\lambda w.xyw))(\lambda w.xyw) & \approx_{dB} & (\lambda.\lambda.320)(\lambda.210) \end{array}$$

Verifiquemos ahora la reducción anónima; se dan todos los pasos por claridad:

$$\begin{array}{rcl}
\big(\lambda.(\lambda.1)0\big)(\lambda.210) & \to_\beta & \mathsf{shift}\,\big(-1,0,((\lambda.1)0)[0 := \mathsf{shift}(1,0,\lambda.210)]\big) \\
& = & \mathsf{shift}\,\big(-1,0,((\lambda.1)0)[0 := \lambda.\,\mathsf{shift}(1,0+1,210)]\big) \\
& = & \mathsf{shift}\,\big(-1,0,((\lambda.1)0)\big[0 := \lambda.\,\mathsf{shift}(1,1,2)\,\mathsf{shift}(1,1,1)\,\mathsf{shift}(1,1,0)\big]\big) \\
& = & \mathsf{shift}\,\big(-1,0,((\lambda.1)0)[0 := \lambda.320]\big) \\
& = & \mathsf{shift}\,\big(-1,0,((\lambda.1)[0 := \lambda.320])(0[0 := \lambda.320])\big) \\
& = & \mathsf{shift}\,\big(-1,0,(\lambda.1[1 := \mathsf{shift}(1,0,\lambda.320)])(\lambda.320)\big) \\
& = & \mathsf{shift}\,\big(-1,0,(\lambda.1[1 := \lambda.\,\mathsf{shift}(1,0+1,320)])(\lambda.320)\big) \\
& = & \mathsf{shift}\,\big(-1,0,(\lambda.1[1 := \lambda.\,\mathsf{shift}(1,1,3)\,\mathsf{shift}(1,1,2)\,\mathsf{shift}(1,1,0)])(\lambda.320)\big) \\
& = & \mathsf{shift}\,\big(-1,0,(\lambda.1[1 := \lambda.430])(\lambda.320)\big) \\
& = & \mathsf{shift}\,\big(-1,0,(\lambda.(\lambda.430))(\lambda.320)\big) \\
& = & \mathsf{shift}\,\big(-1,0,\lambda.(\lambda.430)\big)\,\mathsf{shift}(-1,0,\lambda.320) \\
& = & \big(\lambda.\,\mathsf{shift}(-1,0+1,\lambda.430)\big)\big(\lambda.\,\mathsf{shift}(-1,0+1,320)\big) \\
& = & \big(\lambda.\,\mathsf{shift}(-1,1,\lambda.430)\big)\big(\lambda.\,\mathsf{shift}(-1,1,3)\,\mathsf{shift}(-1,1,2)\,\mathsf{shift}(-1,1,0)\big) \\
& = & \big(\lambda.\,\mathsf{shift}(-1,1,\lambda.430)\big)\big(\lambda.210\big) \\
& = & \big(\lambda.\lambda.\,\mathsf{shift}(-1,1+1,430)\big)\big(\lambda.210\big) \\
& = & \big(\lambda.\lambda.\,\mathsf{shift}(-1,2,4)\,\mathsf{shift}(-1,2,3)\,\mathsf{shift}(-1,2,0)\big)\big(\lambda.210\big) \\
& = & \big(\lambda.\lambda.320)\big)\big(\lambda.210\big) \\
& \approx_{dB} & (\lambda u.(\lambda w.xyw))(\lambda w.xyw)
\end{array}$$

Con lo cual terminamos nuestra exposición acerca de los índices de de-Bruijn, los detalles de la implementación se discutirán en el laboratorio.