Definiendo Números

- Números: especificación

$$\underline{0} \equiv_{def} \ldots$$
$$succ \equiv_{def} (\lambda n.\ \ldots)$$
$$foldNat \equiv_{def} (\lambda s.\lambda z.\lambda n.\ \ldots)$$

cumple que para todos S y Z

$$foldNat\ S\ Z\ \underline{0} \to_{\beta}^{*} Z$$
$$foldNat\ S\ Z\ (succ\ N) =_{\beta} S\ (foldNat\ S\ Z\ N)$$

- Seguimos la idea para otros tipos recursivos (y así obtenemos los “numerales de Church”)

Definiendo Números

- La recursión queda capturada por el *foldNat*
- Cualquier cosa que cumpla esto sirve como números...
- La idea es proceder como con otros tipos recursivos
  - juntar el *foldNat* con los constructores (*foldNat'* = *flip foldNat*)
  - “pasar” los argumentos como parámetros
  - expresar el *foldNat'* como la identidad

Definiendo Números

- Números: solución

$$\underline{0} \equiv_{def} (\lambda s.\lambda z.z)$$
$$succ \equiv_{def} (\lambda n.(\lambda s.\lambda z.s\ (n\ s\ z)))$$
$$foldNat \equiv_{def} (\lambda s.\lambda z.\lambda n.\ n\ s\ z)$$

cumple que para todos S y Z

$$foldNat\ S\ Z\ \underline{0} \to_\beta^* Z$$
$$foldNat\ S\ Z\ (succ\ N) =_\beta S\ (foldNat\ S\ Z\ N)$$

- Nuevamente observamos el "double dispatch" de representar los números como su fold *diferido*

Definiendo Números

- Ejemplos:
  - 2 se representa como $\underline{2}$ dado por *succ* (*succ* $\underline{0}$)), que luego de β-reducir queda (λs.λz.s (s z))
  - $\underline{2} \equiv succ\ (succ\ \underline{0}))$

$$\equiv_{def} \underline{\overbrace{(\lambda n\ s\ z.s\ (n\ s\ z))}^{succ}\overbrace{((\lambda n'\ s'\ z'.s'\ (n'\ s'\ z'))}^{succ}\overbrace{(\lambda s''\ z''.z''))}^{0}}$$

$$\to_{\beta}{}^{*}\ (\lambda s\ z.s\ (\underline{(\lambda n'\ s'\ z'.s'\ (n'\ s'\ z'))(\lambda s'\ 'z''.z'')})\ s\ z)$$

$$\to_{\beta}{}^{*}\ (\lambda s\ z.s\ (\lambda s'\ z'.s'\ (\underline{(\lambda s''\ z''.z'')\ s'\ z'}))\ s\ z)$$

$$\to_{\beta}{}^{*}\ (\lambda s\ z.s\ (\underline{(\lambda s'\ z'.s'\ z')\ s\ z})) \to_{\beta}{}^{*}\ (\lambda s\ z.s\ (s\ z))$$

  - 3 se representa como $\underline{3}$ dado por *succ* (*succ* (*succ* $\underline{0}$))), que luego de β-reducir queda (λs.λz.s (s (s z)))

Definiendo Números

- Más ejemplos
  - 17 se representa como <u>*17*</u> dado por $\underbrace{(\mathit{succ}\ \ldots\ (\mathit{succ}\ \underline{0})\ldots)}_{\text{17 veces}}$,

    que luego de β-reducir queda $(\lambda s.\lambda z.s\ \overbrace{(\ldots\ (s\ z)}\ldots))$
  - un número n se representa como <u>*n*</u> dado por

    $\underbrace{(\mathit{succ}\ \ldots\ (\mathit{succ}\ \underline{0})\ldots)}_{\text{n veces}}$,

    que luego de β-reducir queda $(\lambda s.\lambda z.s\ \overbrace{(\ldots\ (s\ z)}\ldots))$

Definiendo Números

- Notación
  - $F^{(0)}Z \equiv_{\mathrm{def}} Z$
  - $F^{(n+1)}Z \equiv_{\mathrm{def}} F^{(n)}(FZ)$
- Ejemplo:
  - $(\lambda x.x)^{(2)}y$

    $\equiv (\lambda x.x)^{(1)}((\lambda x.x)y)$

    $\equiv (\lambda x.x)^{(0)}((\lambda x.x)((\lambda x.x)y))$

    $\equiv (\lambda x.x)((\lambda x.x)y)$
- Observar:
  - el n en la expresión $F^{(n)}Z$ es una constante fuera de Λ

Definiendo Números

- Observaciones
  - ¡¡los números son funciones!!
  - la cantidad que un 'número' representa se usa para aplicar una función *S* esa cantidad de veces
  - el n utilizado en *n* es una constante fuera de Λ
  - la representación del *0* y la de *False* coinciden
    - (pero no hay problemas, pues no consideramos expresiones en las que no coincidan los 'tipos')

Definiendo Números

- ¿Cómo usamos esta notación para definir cada *n*?
- Usamos el esquema $\boxed{n \equiv_{def} (\lambda s.\lambda z.s^{(n)}z)}$
  - O sea:

$$\underline{0} \equiv_{def} (\lambda s.\lambda z.s^{(0)}z) \equiv_{def} (\lambda s.\lambda z.z)$$

$$\underline{1} \equiv_{def} (\lambda s.\lambda z.s^{(1)}z) \equiv_{def} (\lambda s.\lambda z.s\ z)$$

$$\underline{2} \equiv_{def} (\lambda s.\lambda z.s^{(2)}z) \equiv_{def} (\lambda s.\lambda z.s\ (s\ z))$$

$$\vdots$$

Definiendo Números

- ¿Cómo definimos funciones sobre naturales?
  - Con *foldNat*
- Ejemplo:
  - definir un término *suma* para la función suma
  - debe cumplir *suma* <u>*n*</u> <u>*m*</u> $\rightarrow_\beta^*$ <u>*n+m*</u>
  - podemos usar *foldNat*
    - *suma* $\equiv_{def}$ (λn.λm.*foldNat succ* m n)

Definiendo Números

- Vemos que *succ* se usa de la siguiente manera
  - $m+n$ es igual a sumar n veces 1 a m (o sea, $succ^{(n)}m$)
  - $succ\ (succ\ (succ\ \ldots\ (succ\ m)\ \ldots\ ))$
- Después de reducir queda

$$\boxed{suma \equiv_{def} (\lambda n.\lambda m.n\ succ\ m)}$$

Definiendo Números

- Ejercicios
  - definir un término para representar la multiplicación
  - definir un término *isNotZero*, que cumpla
    - *isNotZero* $\underline{0}$ $\rightarrow_{\beta}{}^{*}$ *False*
    - *isNotZero* $\underline{n+1}$ $\rightarrow_{\beta}{}^{*}$ *True*
  - definir términos
    - *isZero*, para la función que dice si un número es 0
    - *exp*, para representar la exponenciación
    - *pred*, para representar la función que resta uno (difícil)
    - *resta*, para representar la resta de dos naturales


FIN
DESVIO