

Cálculo de objetos

Ejercicio 13

Decir si los siguientes pares de términos definen al mismo objeto o no. Justificar.

1. $o_1 \equiv [\arg = \varsigma(x)x.\arg, \text{val} = \varsigma(x)x.\arg]$ y $o_2 \equiv [\text{val} = \varsigma(z)z.\arg, \arg = \varsigma(v)v.\arg]$
 - Definen el mismo objeto: responden los mismos mensajes y los métodos asociados son los iguales (son α -equivalentes).
2. $o_3 \equiv [\arg = \varsigma(x)x.\arg, \text{val} = \varsigma(x)x.\arg]$ y $o_4 \equiv [\text{foo} = \varsigma(z)z.\arg, \arg = \varsigma(v)v.\arg]$
 - No definen el mismo objeto: no responden los mismos mensajes.

Ejercicio 14

Considerar $o \equiv [\arg = \varsigma(x)x, \text{val} = \varsigma(x)x.\arg]$. Derivar utilizando las reglas de la semántica operacional las reducciones para las siguientes expresiones:

1. $o.\text{val}$

$$\frac{\frac{\frac{}{o \rightarrow o}[\text{obj}]}{\frac{}{o \rightarrow o}[\text{obj}]} \quad \frac{\frac{}{x\{o/x\} = o \rightarrow o}[\text{obj}]}{\frac{}{x.\arg\{o/x\} = o.\arg \rightarrow o}[\text{sel}]}}{\frac{}{o.\text{val} \rightarrow o}[\text{sel}]}$$

2. $o.\text{val}.\arg$

$$\frac{\frac{\frac{}{o \rightarrow o}[\text{obj}]}{\frac{}{o \rightarrow o}[\text{obj}]} \quad \frac{\frac{}{x\{o/x\} = o \rightarrow o}[\text{obj}]}{\frac{}{x.\arg\{o/x\} = o.\arg \rightarrow o}[\text{sel}]} \quad \frac{\frac{}{o \rightarrow o}[\text{obj}]}{\frac{}{o \rightarrow o}[\text{obj}]} \quad \frac{\frac{}{x\{o/x\} = o \rightarrow o}[\text{obj}]}{\frac{}{o.\arg \rightarrow o}[\text{sel}]}}{\frac{}{o.\text{val}.\arg \rightarrow o}[\text{sel}]}$$

3. $(o.\arg \leftarrow \varsigma(z)0).\text{val}$

$$\frac{\frac{\frac{}{o \rightarrow [\arg = \varsigma(x)x, \text{val} = \varsigma(x)x.\arg]}[\text{obj}]}{\frac{}{(o.\arg \leftarrow \varsigma(z)0) \rightarrow o' \quad o' \equiv [\arg = \varsigma(z)0, \text{val} = \varsigma(x)x.\arg]}[\text{upd}]} \quad \frac{\frac{}{o' \rightarrow [\arg = \varsigma(z)0, \text{val} = \varsigma(x)x.\arg]}[\text{obj}]}{\frac{}{\theta[o'/z] = 0 \rightarrow 0}[\text{sel}]} \quad \frac{}{x.\arg\{o'/x\} = o'.\arg \rightarrow 0}[\text{sel}]}{\frac{}{(o.\arg \leftarrow \varsigma(z)0).\text{val} \rightarrow 0}[\text{sel}]}$$

PREGUNTA: En $0 \rightarrow 0$ también se usa la regla obj?

Ejercicio 15

Sea $o \equiv [a = \varsigma(x)(x.a \leftarrow \varsigma(y)(y.a \leftarrow \varsigma(z)[[]]))]$. Mostrar cómo reduce $o.a.a$.

$$\frac{\frac{\frac{}{o \rightarrow [a = \varsigma(x)(x.a \leftarrow \varsigma(y)(y.a \leftarrow \varsigma(z)[[]]))]}[\text{obj}]}{\frac{}{o.a \rightarrow o' \quad o' \equiv [a = \varsigma(y)(y.a \leftarrow \varsigma(z)[[]]))]}[\text{upd}]} \quad \frac{\frac{}{o' \rightarrow [a = \varsigma(y)(y.a \leftarrow \varsigma(z)[[]]))]}[\text{obj}]}{\frac{}{(y.a \leftarrow \varsigma(z)[[]])\{o'/y\} = (o'.a \leftarrow \varsigma(z)[[]]) \rightarrow o'' \quad o'' \equiv [a = \varsigma(z)[[]]]}[\text{upd}]} \quad \frac{}{(1)}[\text{sel}]}{\frac{}{o.a.a \rightarrow [a = \varsigma(z)[[]]]}[\text{sel}]}$$

Ejercicio 16

1. Definir true y false como objetos con los siguientes tres métodos: not, if, e ifnot. Notar que tanto if como ifnot deberán retornar una función binaria. Las operaciones deberían satisfacer las siguientes igualdades:

- `true.not = false`
- `false.not = true`
- `true.if v1 v2 = v1`
- `true.ifnot v1 v2 = v2`
- `false.if v1 v2 = v2`
- `false.ifnot v1 v2 = v1`

`true` \equiv `[not = ς (t)[not = t, if = t.ifnot, ifnot = t.if], if = λ (b1) λ (b2)(b1), if = λ (b1) λ (b2)(b2)]`

`false` \equiv `true.not`

PREGUNTA: vale pedirle `true.if` sin pasarle los parámetros (para obtener la función nada más)?
se puede definir así: `not = sigma(t)((not \leftarrow t).if \leftarrow t.ifnot).ifnot \leftarrow t.if?`

2. Definir `and` y `or` como objetos que se comporten como funciones que esperen dos argumentos (para poder escribir expresiones como `and(true)(false)`, etc).

`and` \equiv `λ (x) λ (y)x.if(y)(x)` PREGUNTA: esto es correcto o hay que escribir `[[λ x. λ y. x.if(y)(false)]]?`

`or` \equiv `λ (x) λ (y)x.if(x)(y)`

Ejercicio 17

1. Definir el objeto `origen` que representa el origen de coordenadas en dos dimensiones. Este objeto provee tres operaciones: los observadores `x` y `y` y `mv` tal que `origen.mv v w` desplaza a `origen v` unidades a la derecha y `w` unidades hacia arriba.

`origen` \equiv `[x = 0, y = 0, mv = ς (o) λ (v) λ (w)(o.x \leftarrow o.x+v).y \leftarrow o.y+w]`

2. Definir una clase `Punto`, cuyas instancias proveen las operaciones `x`, `y` y `mv`. Considerar que los puntos se crean con sus coordenadas en 0.

```
Punto = [
  new =  $\varsigma$ (c)[ coordenadas =  $\varsigma$ (i)c.origen(i), x =  $\varsigma$ (i)c.x(i), y =  $\varsigma$ (i)c.y(i), mv =  $\varsigma$ (i)c.mv(i) ]
  origen =  $\lambda$ (i) origen,
  x =  $\lambda$ (i) i.coordenadas.x,
  y =  $\lambda$ (i) i.coordenadas.y,
  mv =  $\lambda$ (i) $\lambda$ (v) $\lambda$ (w) i.coordenadas  $\leftarrow$  i.coordenadas.mv(v)(w)
]
```

PREGUNTA: puedo hacer que el método de clase agarre tantos parámetros como quiera?

3. Mostrar como reduce `Punto.new`.

```
[obj]
Punto  $\rightarrow$  Punto [coordenadas =  $\varsigma$ (i)Punto.origen(i), x =  $\varsigma$ (i)Punto.x(i), y =  $\varsigma$ (i)Punto.y(i), mv =  $\varsigma$ (i)Punto.mv(i)] = ins  $\rightarrow$  ins [obj]
Punto.new  $\rightarrow$  ins [sel]
```

4. Definir la subclase `PuntoColoreado`, que permite construir instancias de puntos que tienen asociado un color. Además del método `new`, que crea puntos blancos, la clase debe contar con un método que cree puntos de un color pasado como parámetro.

```
PuntoColoreado = [
  new =  $\varsigma$ (c)[ coordenadas =  $\varsigma$ (i)c.origen(i), x =  $\varsigma$ (i)c.x(i), y =  $\varsigma$ (i)c.y(i), mv =  $\varsigma$ (i)c.mv(i), color =  $\varsigma$ (i)c.color(i) ]
  newConColor =  $\varsigma$ (c) $\lambda$ (color)c.new.color  $\leftarrow$  color
  origen = Punto.origen,
  x = Punto.x,
  y = Punto.y,
  mv = Punto.mv,
  color =  $\lambda$ (i)blanco
]
```

Ejercicio 18

1. Se desea definir a los números naturales como objetos de manera análoga al ejercicio 3. Estos objetos deben proveer las operaciones `esCero` y `succ` que permiten respectivamente testear si el receptor del mensaje es 0 o no, y obtener al sucesor del receptor. Además, todos los números distintos de 0 deben proveer la operación `pred`.

`cero` = `[esCero = true, succ = ς (n)(n.esCero \leftarrow false).pred = n]`

2. Definir `iguales` y `menor` como objetos que se comporten como funciones que esperen dos argumentos.

```
iguales = λ(x)λ(y)(
    or(
        and(x.esCero)(y.esCero))
        (x.esCero.if
            (false)
            (y.esCero.if
                (false)
                (iguales
                    (x.pred)
                    (y.pred)
                )
            )
        )
    )
)
```