DEFINICIONES LOCALES Y PATRONES

Agregamos al lenguaje la posibilidad de definir localmente y notación para patrones:

<exp> ::= let <pat> ≡ <exp>, ..., <pat> ≡ <exp> in <exp>
      | λ<pat>.<exp>
<pat> ::= <var> | <<pat>,..,<pat>>

Así podemos escribir λ<u,<v,w>>.u v w en vez de λt. t.0 t.1.0 t.1.1

Para no tener que definir evaluación y semántica denotacional eager y normal para esta extensión, nos conformamos con ver que estas nuevas expresiones se pueden definir en términos de las que ya existían. Es sólo azúcar sintáctico:

λ<p1,..,pn>.e es lo mismo que λv.let p1 ≡ v.0, .., pn ≡ v.[n-1] in e

donde v es una variable nueva (no ocurre libre en e ni en ninguno de los patrones)

let p1 ≡ e1, .., pn ≡ en in e es lo mismo que (λp1..λpn.e) e1 .. en

Aplicando repetidamente estas dos transformaciones podemos eliminar los patrones que no sean variables y las definiciones locales (let) obteniendo una expresión cuya semántica ya está definida.

Tener en cuenta que cuando n = 0, let p1 ≡ e1, .., pn ≡ en in e quedaría let in e, esto en realidad es directamente la expresión e (el let in sin nada al medio es como si no existiera).

RECURSIÓN

Recursión ha adoptado notaciones diferentes en las versiones eager y normal: en la eager se utiliza el letrec y en la normal, rec:

<exp> ::= letrec <var> ≡ λ<var>.<exp>, ..., <var> ≡ λ<var>.<exp> in <exp>
      | rec <exp>

El letrec permite hacer definiciones recursivas como

letrec fact ≡ λn.if n = 0 then 1 else n * fact (n-1) in fact 10

mientras que rec se utiliza como el operador de punto fijo:

rec (λf.λn.if n = 0 then 1 else n * f (n-1)) 10

Evaluación Eager:

$$\frac{e'/(v \to \lambda u.\text{letrec } v \equiv \lambda u.e \text{ in } e) \Rightarrow z}{\text{letrec } v \equiv \lambda u.e \text{ in } e' \Rightarrow z} \; v \neq u,$$

Evaluación Normal:

$$\frac{e\ (\text{rec } e) \Rightarrow z}{\text{rec } e \Rightarrow z}$$

Semántica Denotacional Eager:

[[letrec v ≡ λu.e in e']]η = [[e']]η'

donde

η' = [η|v:[[λu.e]]η']

pero la definición de η' está mal tipada, porque [[λu.e]]η' pertenece a D, no puede estar en el ambiente que es una función de <var> en V. Se reescribe

η' = [η|v:Lfun (λz∈V.[[e]][η'|u:z])]

que está bien tipada. Se logró resolver gracias a que el definiendo en el letrec es una abstracción.
El problema ahora es que η' está definido en términos de sí mismo. Al no ser Env un dominio, no podemos aplicar el teorema del menor punto fijo.

Se reescribe

η' = [η|v:Lfun f]
f = (λz∈V.[[e]][η|v:Lfun f|u:z])

Ahora la f se definió en términos de sí mismo, pero se puede aplicar el teorema del menor punto fijo porque Vfun es un dominio.

Semántica Denotacional Normal:

$[[\text{rec } e]]\eta = (\lambda f \in Vfun.Y\ f)fun^*\ ([[e]]\eta)$

donde Y es el operador de menor punto fijo, $Y\ f = \sup f^i \perp$.