Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Licenciatura en Ciencias de la Computación
Análisis de Lenguajes de Programación

Programación Monádica

Alumnos:
CRESPO, Lisandro (C-6165/4)
MISTA, Agustín (M-6105/1)

Docentes:
JASKELIOFF, Mauro
SIMICH, Eugenia
MANZINO, Cecilia
RABASEDAS, Juan Manuel

05 de Noviembre de 2015

Ejercicio 1.a *Demostramos que `State` es una mónada:*

Probamos que verifique las tres leyes de las mónadas.
Comenzamos probando `return x` $\gg=$ `f = f x`

$$
\begin{aligned}
& return\ x \gg= f \\
= & \langle \text{ def. } \texttt{return} \rangle \\
& State\ (\lambda s \rightarrow (v, s')) \gg= f \\
= & \langle \text{ def. } >>= \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = runState\ (State\ (\lambda s \rightarrow (x, s))\ s\ in\ runState\ (f\ v)\ s') \\
= & \langle \text{ def. runState } \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = (\lambda s \rightarrow (x, s))\ s\ in\ runState\ (f\ v)\ s') \\
= & \langle \text{ aplicación } \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = (x, s)\ in\ runState\ (f\ v)\ s') \\
= & \langle \text{ let } \rangle \\
& State\ (\lambda s \rightarrow runState\ (f\ x)\ s) \\
= & \langle\ \eta\text{-equivalencia } \rangle \\
& State\ (runState\ (f\ x)) \\
= & \langle \text{ State} \circ \text{runState} = \text{id } \rangle \\
& id\ (f\ x) \\
= & \langle \text{ def. id } \rangle \\
& f\ x
\end{aligned}
$$

Continuamos probando `m` $\gg=$ `return = m`. Lo hacemos por inducción estructural sobre el tipo de datos `State`.

$$
\begin{aligned}
& m \gg= return \\
= & \langle \text{ def. } >>= \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = runState\ m\ s\ in\ runState\ (return\ v)\ s') \\
= & \langle \text{ def. return } \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = runState\ m\ s\ in\ runState\ (State\ (\lambda s \rightarrow (v, s)))\ s') \\
= & \langle \text{ def. runState } \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = runState\ m\ s\ in\ (\lambda s \rightarrow (v, s))\ s') \\
= & \langle \text{ aplicación } \rangle \\
& State\ (\lambda s \rightarrow let\ (v, s') = runState\ m\ s\ in\ (v, s')) \\
= & \langle \text{ let } \rangle \\
& State\ (\lambda s \rightarrow runState\ m\ s) \\
= & \langle \text{ aplicación } \rangle \\
& State\ (runState\ m) \\
= & \langle \text{ State} \circ \text{runState} = \text{id } \rangle \\
& id\ m \\
= & \langle \text{ def. id } \rangle \\
& m
\end{aligned}
$$

Por último, probamos `(t ≫= f) ≫= g = t ≫= (λx → f x ≫= g)`.

$(t \gg= f) \gg= g$
$= \langle \text{ def } >>= \rangle$
$State\ (\lambda s \to let\ (v, s') = runState\ (t \gg= f)\ s\ in\ runState\ (g\ v)\ s')$
$= \langle \text{ def } >>= \rangle$
$State\ (\lambda s \to let\ (v, s') = runState\ (State\ (\lambda s \to let\ (v, s') = runState\ t\ s\ in\ runState\ (f\ v) s'))\ s\ in\ runState\ (g\ v)$
$= \langle \text{ def runState } \rangle$
$State\ (\lambda s \to let\ (v, s') = (\lambda s \to let\ (v, s') = runState\ t\ s\ in\ runState\ (f\ v) s'))\ s\ in\ runState\ (g\ v)\ s')$
$= \langle \text{ aplicación } \rangle$
$State\ (\lambda s \to let\ (v, s') = let\ (v, s') = runState\ t\ s\ in\ runState\ (f\ v)\ s')\ in\ runState\ (g\ v)\ s')$
$= \langle \text{ introducción State } (f\ v :: State\ s \to \exists\ f'\ / f\ v\ = \ State\ f') \rangle$
$State\ (\lambda s \to let\ (v, s') = let\ (v, s') = runState\ t\ s\ in\ runState\ (State\ f')\ s')\ in\ runState\ (g\ v)\ s')$
$= \langle \text{ aplicación } \rangle$

$((St\ h) >>= f) >>= g$
$= \langle \text{ definición } >>= \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ runState\ (f\ x)\ s') >>= g$
$= \langle \text{ definición } >>= \rangle$
$St\ (\lambda t \to let\ (y, t') = (\lambda s \to let\ (x, s') = h\ s\ in\ runState\ (f\ x)\ s')\ t\ in\ runState\ (g\ y)\ t')$
$= \langle \text{ aplicación } \rangle$
$St\ (\lambda t \to let\ (y, t') = (let\ (x, s') = h\ t\ in\ runState\ (f\ x)\ s')\ in\ runState\ (g\ y)\ t')$
$= \langle \text{ f x :: State s a} \Rightarrow \exists f'\ /\ f\ x = St\ f' \rangle$
$St\ (\lambda t \to let\ (y, t') = (let\ (x, s') = h\ t\ in\ runState\ (St\ f')\ s')\ in\ runState\ (g\ y)\ t')$
$= \langle \text{ runState} \circ \text{St} = \text{id } \rangle$
$St\ (\lambda t \to let\ (y, t') = (let\ (x, s') = h\ t\ in\ f'\ s')\ in\ runState\ (g\ y)\ t')$
$= \langle \text{ propiedad de let } \mathbf{??} \rangle$
$St\ (\lambda t \to let\ (x, s') = h\ t\ in\ (let\ (y, t') = f'\ s'\ in\ runState\ (g\ y)\ t'))$
$= \langle\ \alpha\text{-equivalencia } \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ (let\ (y, t') = f'\ s'\ in\ runState\ (g\ y)\ t'))$
$= \langle \text{ abstracción } \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ (\lambda t \to let\ (y, t') = f'\ t\ in\ runState\ (g\ y)\ t')\ s')$
$= \langle \text{ runState} \circ \text{St} = \text{id } \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ runState(St(\lambda t \to let\ (y, t') = f'\ t\ in\ runState\ (g\ y)\ t'))\ s')$
$= \langle \text{ definición } >>= \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ runState\ (St\ f' >>= g)\ s')$
$= \langle\ St\ f' = f\ x\ \mathbf{??} \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ runState\ (f\ x >>= g)\ s')$
$= \langle \text{ aplicación } \rangle$
$St\ (\lambda s \to let\ (x, s') = h\ s\ in\ runState\ ((\lambda x \to f\ x >>= g)\ x)\ s')$
$= \langle \text{ definición } >>= \rangle$
$(St\ h) >>= (\lambda x \to f\ x >>= g)$

CRESPO, Lisandro

MISTA, Agustín